SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Tihomir Katić

OPTIMIRANJE SIGURNOSNIH PRAVILA VATROZIDA

MAGISTARSKI RAD

Zagreb, 2011.

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING
SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Tihomir Katić

OPTIMISATION OF FIREWALL SECURITY RULES
OPTIMIRANJE SIGURNOSNIH PRAVILA VATROZIDA

MASTER THESIS
MAGISTARSKI RAD

Zagreb, 2011.

MAGISTARSKI RAD JE IZRAĐEN JE NA ZAVODU ZA ELKTRONIČKE SUSTAVE I OBRADBU NFORMACIJA

FAKULTETA ELEKTROTEHNIKE I RAČUNARSTVA

MENTOR:
doc. dr. sc. Mile Šikić (Fakultet elektrotehnike i računarstva, Zagreb)

Magistarski rad ima: 123 strana

Magistarski rad br.:_____

SADRŽAJ

1. Uvod

Za svaki mrežni paket koji napušta ili ulazi u računalnu mrežu vatrozid treba odlučiti hoće li taj paket prihvatiti, odbaciti ili poduzeti neku drugu akciju (logiranje, promjena podataka u paketu, preusmjeravanje i dr.). Što se prije ta odluka donese to su performanse vatrozida bolje, a paket prije procesuiran. Ovaj rad predstavlja jedan pristup optimiranja sigurnosnih pravila za potrebe poboljšanja vatrozidnih performansi.

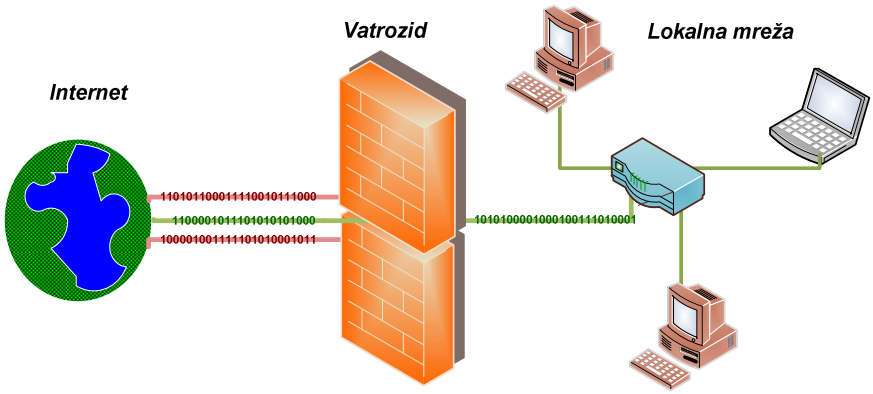
Dosadašnji radovi na temu optimiranja sigurnosnih pravila vatrozida zasnovani su na osnovnim elementima mrežnih paketa (izvorna i ciljna IP adresa, izvorni i ciljni port, protokol) i to s pojednostavljenim mogućim vrijednostima (bez invertiranih vrijednosti, bez korištenje IP *wildcard*[1] mrežnih usporedbi, bez korištenja niza mogućih vrijednosti kod portova ili raspona IP adresa). Također, baziraju se na samo dvije moguće akcije (prihvaćanje ili odbacivanje). Iako ti elementi jesu najčešće korišteni u sigurnosnim pravilima, oni nisu jedini. Iz tog razloga su u programskoj implementaciji napravljenoj na temelju ovog rada – Firo (*FIRewall Optimizer*) podržani svi parametri koji se mogu naći u Iptables programskom paketu (preko 100 parametara usporedbe), sve njihove moguće vrijednosti, kao i sve moguće akcije (27 standardnih akcija – npr. prihvaćanje, odbacivanje, logiranje, NAT preusmjeravanja, klasificiranje i dr.) te proizvoljni broj korisnički definiranih preusmjeravanja. Osim toga, implementirana optimizacijska tehnika uvodi optimiranje parametara sigurnosnih pravila, tj. uklanjanje nepotrebnih parametara unutar jednog pravila (tzv. mikro-optimiranje); a uz to i optimiranje pravila logiranja njihovim premještanjem neposredno prije relevantnih akcija. Ukoliko se pojave novi parametri ili nove akcije, program je jednostavno proširiti. Iako je kao ulaz korištena Iptables sintaksa, uz manje prilagodbe Firo se može koristiti i za druge vatrozide.

U sljedećem poglavlju 2 (Sigurnosna pravila vatrozida) ukratko su opisani vatrozidi te sigurnosna pravila i njihovi mogući odnosi koji su važni za utvrđivanje redundancija među njima. Potom je u poglavlju 3 (Anomalije u sigurnosnim pravilima) dana uobičajena klasifikacija anomalija koja je nadograđena novim anomalijama. U poglavlju 4 (Metode optimiranja poretka sigurnosnih pravila) je pregled postojećih metodologija optimiranja vatrozida. Budući da je Firo temeljen na Iptables sigurnosnim naredbama, u poglavlju 5 (Sigurnosna pravila - Iptables vatrozid) opisane su podržane opcije tih naredbi. A u poglavlju 6 (Firo – optimizator sigurnosnih pravila) detaljno su navedene funkcionalnosti razvijenog programa i algoritmi koji se koriste te je dana osnovna analiza učinkovitosti i plan budućeg razvoja.

[1] *Wildcard* – niz bitova koji omogućuje označavanje IP adresa koje nisu dio neprekinutog niza, ali zadovoljavaju definirani uzorak (npr. 10.0.0.0/255.0.0.1 označava sve adrese koje završavaju s parnim brojem u nizu 10.0.0.0-10.255.255.255)

2. Sigurnosna pravila vatrozida

Kako bi zaštitili svoje računalne mreže organizacije sve više vremena i resursa posvećuju računalnoj sigurnosti. Važna i nezaobilazna komponenta te računalne sigurnosti su vatrozidi (engl. *firewall*), računalne komponente koje mogu služiti za zaštitu osobnih računala (tzv. osobni vatrozidi) ili za zaštitu većeg broj računala smještenih u jednoj ili više mreža (tzv. mrežni vatrozidi).



Slika 1: Prikaz zaštite lokalne mreže pomoću mrežnog vatrozida

U svom radu ispravno konfigurirani vatrozidi minimaliziraju računalne sigurnosne prijetnje, koje mogu biti u rangu od znatiželjnih pokušaja neupućenih Internet korisnika do vrlo organiziranih i tehnički dobro pripremljenih zlonamjernih napadača koji bi u slučaju uspješno provedenog napada mogli dobiti pristup privatnim informacijama ili utjecati na ispravno korištenje sustava od strane ovlaštenih korisnika.

Osnovna komponenta vatrozida su sigurnosna pravila koja mogu određene oblike mrežnog prometa ili dozvoljavati i propuštati ih prema njihovom odredištu, ili ih mogu zabranjivati čime im se onemogućava prometovanje prema ciljanim odredištima. Pri tome različite organizacije imaju različite potrebe za dozvoljenim mrežnim prometom te stoga kreiraju različite nizove sigurnosnih pravila.

Sigurnosno pravilo u sebi sadrži elemente koji definiraju mrežne pakete na koja se to pravilo primijenjuje. Ti elementi nazivaju se parametrima uspoređivanja (mrežnog paketa s pravilom). Osim parametara uspoređivanja, sigurnosno pravilo sadrži i akciju. Standardne akcije su prihvaćanje ili odbacivanje mrežnog paketa, ali akcije mogu biti i npr. preusmjeravanje mrežnog paketa, skrivanje privatne IP adrese, označavanje paketa na sustavu, statističke akcije i sl. Kod tih nestandardnih akcija pojavljuju se i akcijski parametri koji čine cjelinu s akcijom (npr. kod preusmjeravanja definira se nova ciljna adresa).

2.1. Redundantnost sigurnosnih pravila

Što je veći broj različitih politika koje se primjenjuju za različite oblike mrežnog prometa to je veći i broj sigurnosnih pravila na vatrozidima. U slučajevima kad je broj sigurnosnih pravila na vatrozidu velik npr. nekoliko stotina ili tisuća različitih pravila, administratori u pravilu nisu u mogućnosti efikasno kreirati nova i uklanjati stara sigurnosna pravila. Čak je i vrlo iskusnim administratorima kompliciranо uspoređivati nova pravila sa svim postojećima te pronalaziti moguće redundancije. Vjerojatnost otkrivanja redundantnosti je još manja ukoliko organizacija promjeni administratora zaduženog za konfiguraciju vatrozida. U slučaju da je administrator neiskusan, za očekivati je veći broj redundantnih sigurnosnih pravila na vatrozidu.

Redundantna sigurnosna pravila su ona pravila koja se neće nikad primijeniti ili će se nepotrebno duplo primijeniti. Npr. ta pravila nikad neće prepoznati određeni oblik mrežnog prometa jer im prethode neka druga pravila koja uspoređuju isti mrežni promet koji uspoređuju i redundantna pravila. Stoga sav mrežni promet biva ili odbačen ili propušten. Taj odbačeni ili propušteni mrežni promet ne predstavlja problem za performanse vatrozida jer se on nakon odbacivanja ili propuštanja više ne uspoređuje s ostalim sigurnosnim pravilima, ali sav ostali promet za koji postoje pravila koja se nalaze iza redundantnog pravila ili mrežni promet za koji uopće ne postoje definirana pravila, nepotrebno se uspoređuje s redundantnim pravilom te se tako umanjuju performanse vatrozida.

Do narušavanja performansi vatrozida dolazi i kad je sigurnosno pravilo nepotrebno podijeljeno na dva ili više sigurnosnih pravila. U tim slučajevima mrežni promet koji se mogao uspoređivati sa samo jednim pravilom se uspoređuje s dva ili više sigurnosnih pravila, a ako se između njih nalaze i druga pravila, uspoređuje se i s njima.

Kako bi se otkrile i uklonile redundancije bitno je utvrditi moguće odnose između sigurnosnih pravila što je definirano u nastavku.

2.2. Odnosi između sigurnosnih pravila

E. S. Al-Shaer i H. H. Hamed [6], [9] definirali su pet mogućih odnosa između sigurnosnih pravila. Pri tome su postavili ograničenje jer je u definiranju odnosa uključeno samo 5 parametara uspoređivanja (protokol, izvorna i ciljna adresa, izvorni i ciljni port). Ta pravila su korektna, ali je za potrebe ovog rada u nastavku 5 definiranih parametara prošireno sa svim dostupnim parametrima usporedbe sigurnosnih pravila koji služe za identificiranje mrežnog prometa. Osim toga, za potrebe ovog rada definiran je i jedan novi odnos.

Definicija 1: Pravila R_x i R_y su u *potpunosti razdvojena* ako svaki parametar iz pravila R_x nije jednak i nije podskup i nije nadskup pripadajućeg parametra iz pravila R_y. Formalno:

$R_x \ \Re_{PR} \ R_y$ ako i samo ako:

$\forall i$: $R_x[i] \not\bowtie R_y[i]$,

gdje su $i, j \in$ { parametri usporedbe}, $\bowtie \in \{\subset, \supset, =\}$

U slučaju kad se ne promatra samo 5 najčešćih parametara pravila (protokol, izvorna i ciljna adresa, izvorni i ciljni port) nerealno je očekivati postojanje sigurnosnih pravila kojima su svi mogući parametri potpuno razdvojeni, jer bi svi takvi parametri morali biti navedeni, naravno uz određena ograničenja vezana uz protokol (npr. nije moguće definirati ICMP broj uz TCP protokol), a ideja ovog rada je omogućiti jednostavno dodavanje novih parametara, čak i bez konfiguracije.

Definicija 2: Pravila R_x i R_y se u *potpunosti preklapaju* ako je svaki parametar iz pravila R_x jednak pripadajućem parametru iz pravila R_y. Formalno:

$R_x \ \Re_{PP} \ R_y$ ako i samo ako:

$\forall i$: $R_x[i] = R_y[i]$,

gdje je $i \in$ { parametri usporedbe}

Tablica 1: Primjer potpuno preklopljenih sigurnosnih pravila

Pravilo	Protokol	Izvorna adresa	ICMP tip	TTL vrijednost	Akcija
R1	ICMP	1.0.*.*	8	254	Logiraj
R2	ICMP	1.0.*.*	8	254	Prihvati

Definicija 3: Pravila R_x i R_y se *uključivo preklapaju* ako se ne preklapaju u potpunosti, a svaki parametar iz pravila R_x je jednak ili je podskup pripadajućeg parametra iz pravila R_y. Formalno:

$R_x \ \Re_{UP} \ R_y$ ako i samo ako:

$\forall i$: $R_x[i] \subseteq R_y[i]$,

$\exists j$: $R_x[j] \neq R_y[j]$,

gdje su $i, j \in$ { parametri usporedbe}

Tablica 2: Primjer uključivo preklopljenih sigurnosnih pravila

Pravilo	Protokol	Ciljna adresa	Ciljni port	TCP zastavice	Akcija
R1	TCP	1.0.1.1	1024:65535	SYN=1,ACK=0	Odbaci
R2	TCP	1.0.*.*	0:65535	SYN=1,ACK=0	Prihvati

Definicija 4: Pravila R_x i R_y su *djelomično razdvojena* ako postoji barem jedan parametar iz pravila R_x koji nije jednak i nije podskup i nije nadskup

pripadajućeg parametra iz pravila R_y, te postoji barem jedan parametar iz pravila R_x koji je ili jednak ili je podskup ili je nadskup pripadajućeg parametra iz pravila R_y Formalno:

$R_x \Re_{DR} R_y$ ako i samo ako:

$\exists\ i$: $R_x[i] \bowtie R_y[i]$,

$\exists\ j$: $R_x[j] \not\bowtie R_y[j]$,

gdje su $i, j \in$ { parametri usporedbe}, $\bowtie \in \{\subset, \supset, =\}$

Tablica 3: Primjer djelomično razdvojenih sigurnosnih pravila

Pravilo	Protokol	TOS	Duljina paketa	Ulazno sučelje	Akcija
R1	UDP	Maximize-Reliability	100:200	eth0	Preusmjeri
R2	TCP	Minimize-Cost	100:1000	eth1	Prihvati

Definicija 5: Pravila R_x i R_y su *korelirana* ako su neki parametri iz R_x podskup ili jednaki pripadajućim parametrima iz R_y, dok su ostali parametri iz R_x nadskup pripadajućim parametrima iz R_y. Formalno:

$R_x \Re_K R_y$ ako i samo ako:

$\forall\ i$: $R_x[i] \bowtie R_y[i]$,

$\exists\ j$: $R_x[i] \subset R_y[i]$,

$\exists\ k$: $R_x[k] \supset R_y[k]$,

gdje su $i, j, k \in$ { parametri usporedbe}, $\bowtie \in \{\subset, \supset, =\}$

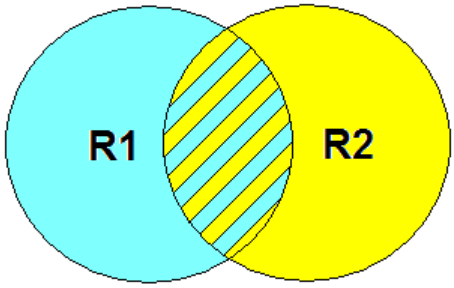
Tablica 4: Primjer koreliranih sigurnosnih pravila

Pravilo	Protokol	Ulazno sučelje	Izlazno sučelje	Duljina paketa	Akcija
R1	Svi	eth0	Svi	100:200	Odbaci
R2	TCP	Svi	eth1	100:200	Prihvati

Navedenih 5 definicija ne pokrivaju sve odnose između sigurnosnih pravila. To je vidljivo na primjeru koji prikazuje Tablica 5 gdje dva definirana pravila zadovoljavaju jedino definiciju 4 djelomično razdvojenih pravila. Međutim, ni ta definicija nije dovoljno jasna jer ona može ukazivati na dva pravila koja dijelom pokrivaju isti mrežni promet (Tablica 5), ali i na pravila koja uopće ne pokrivaju isti mrežni promet (Tablica 3). E. S. Al-Shaer i H. H. Hamed to su riješili na način da postojeća pravila razdvajaju na više manjih pravila pri čemu presjek njihovih parametara nikad ne smije biti neki novi parametar. Npr. ukoliko postoji jedno pravilo s rasponom portova 10:20 i drugo pravilo s rasponom portova 15:25 njihov presjek je raspon 15:20. Stoga se prvo pravilo dijeli na dva pravila koja imaju raspone 10:14 i 15:20, a drugo se dijeli na dva pravila koja imaju raspone 15:20 i 21:25. Oni su mogli na takav način utvrđivati odnose jer su koristili ograničeni skup parametara s jednostavnim vrijednostima, bez inverzije, niza portova i IP *wildcard* maski. Ali kad se koriste kompleksne vrijednosti i nepoznati broj parametara nije realno razdvajati pravila jer bi od jednog pravila ponekad trebalo kreirati i tisuću ili više novih pravila.

Stoga je za potrebe ovog rada bilo nužno definirati dodatni odnos između sigurnosnih pravila koji je u većoj ili manjoj mjeri povezan s prethodnim definicijama, a nazvan je **relacijom**. Ispravno bi bilo i kad bi se Definicija 5 korelacije modificirala, ali da bi se sačuvale te definicije uveden je novi termin.

Dva sigurnosna pravila su u relaciji ukoliko su ta dva pravila definirana dijelom za isti mrežni promet, pri čemu taj zajednički mrežni promet predstavlja presjek mrežnog prometa koji označava prvo pravilo (npr. R1) te mrežnog prometa koji označava drugo pravilo (npr. R2) kao što je to vidljivo na sljedećoj slici:



Slika 2: Relacija dvaju pravila

Definicija 6: Pravila R_x i R_y su u *relaciji* (preklapaju se) ako presjek svakog parametar iz R_x i pripadajućeg parametra iz R_y nije prazan skup. Formalno:

$R_x \ \Re_R \ R_y$ ako i samo ako:

$\forall i$: $R_x [i] \cap R_y [i] \neq \{\}$,

gdje je $i \in$ { parametri usporedbe}

Tablica 5: Primjer preklopljenih sigurnosnih pravila

Pravilo	Protokol	Ulazno sučelje	TCP zastavice	Ciljni port	Akcija
R1	Svi	eth0	SYN=1	100:500	Odbaci
R2	TCP	Svi	SYN=1,ACK=0	22,80,443	Prihvati

Pri svim slučajevima sigurnosnih pravila u relacijama nije svejedno koje pravilo prethodi kojem pravilu kao i da li se druga pravila, koja su u relaciji s njima, nalaze između njih. U tu svrhu, za modeliranje zavisnosti i poretka sigurnosnih pravila, koriste se tzv. direktni aciklički grafovi (engl. DAG – *Directed Acyclical Graphs*) [27].

Relacija između dva sigurnosna pravila postoji kada presjek između svih njihovih parametara postoji i različit je od nule. U slučaju da ne postoji presjek barem jednog parametra tada ti mrežni paketi ne označavaju isti

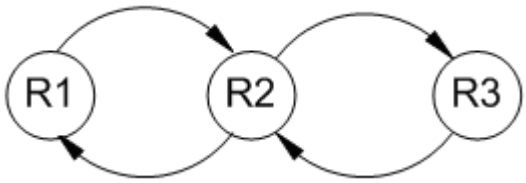
mrežni promet i oni nisu u relaciji. U nastavku je dan primjer sigurnosnih pravila koja se nalaze u relaciji. Potrebno je napomenuti da oni parametri koji se ne navode u sigurnosnom pravilu, posjeduju vrijednost „svi“, tj. obuhvaćaju sve vrijednosti.

Tablica 6: Primjer sigurnosnih pravila u relaciji

Pravilo	Protokol	Izvorna adresa	Ciljna adresa	Ciljni port	Akcija
R1	TCP	1.0.1.*	1.1.1.2	80	Odbaci
R2	ALL	1.0.*.*	1.1.1.2	80	Prihvati
R3	ALL	1.0.2.*	*.*.*.*	0:65535	Odbaci

Ukoliko su dva pravila u relaciji, ali imaju različite akcije (jedno pravilo dozvoljava promet dok ga drugo zabranjuje) tada je važno sačuvati poredak pravila u optimizacijskoj proceduri. U suprotnom se optimirana vatrozidova sigurnosna politika ne bi ponašala kao izvorna. Iz tog razloga je bitno imati reference između povezanih sigurnosnih pravila. I u slučaju kada jedno pravilo zamjenjuje druga ono treba naslijediti sve reference pravila koja zamjenjuje.

Za prethodni primjer iz Tablica 6 od tri naredbe kreira se graf. Prilikom izrade tog grafa potrebno je uočiti kako se sva pravila nalaze u relaciji osim pravila R1 i R3 kojima je presjek izvorne IP adrese jednak nuli.



Slika 3: Zavisnosti triju sigurnosnih pravila

3. Anomalije u sigurnosnim pravilima

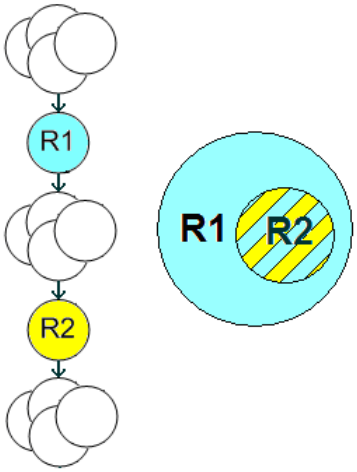
Najčešće se u stručnoj literaturi [9], [18], [20] anomalijama smatraju takve kombinacije sigurnosnih pravila koje označavaju isti mrežni promet. Ta praksa nije u potpunosti slijeđena u ovom dokumentu. Budući da smatram kako termin „anomalije“ označava neku neželjenu situaciju, a brojni su slučajevi gdje je korisno da dva pravila pokrivaju dio istog prometa; u nastavku se za sigurnosna pravila koja označavaju dijelom ili u potpunosti isti mrežni promet koristi prethodno opisani termin **relacije**. A ukoliko se mrežni promet koja dva sigurnosna pravila uspoređuju niti jednom dijelom ne preklapa onda se za ta dva pravila kaže da nisu u relaciji.

Osim što se anomalijama smatraju kombinacije sigurnosnih pravila koje označavaju isti mrežni promet, kao anomalije se klasificiraju i ona sigurnosna pravila što označavaju mrežni promet koji se nikad neće pojaviti na vatrozidu. Uklanjanje ovakvih oblika anomalija koje se nazivaju anomalijama irelevantnosti, zahtijeva poznavanje šire situacije od one koja je vidljiva samo iz vatrozidnih sigurnosnih pravila. Za uklanjanje takvih oblika potrebno je poznavati sve protokole koje pojedine mreže i računala mogu generirati, sve IP adrese koje se mogu pojaviti kao izvorne ili ciljne i sl., ali takva analiza nije obuhvaćena ovim radom.

3.1. Anomalija zasjenjenja

Originalno [6] anomalija zasjenjenja (engl. *shadowing*) predstavlja takvu kombinaciju sigurnosnih pravila pri čemu prvo pravilo označava sav mrežni promet koji označava i drugo pravilo tj. mrežni promet koji označava drugo pravilo podskup je mrežnog prometa označenog prvim pravilom. Prema već spomenutim Definicijama 1 i 2 može se reći kako se drugo pravilo uključivo ili potpuno preklapa s prvim pravilom. Prvo pravilo se pri tome nalazi u poretku svih pravila ispred drugog pravila. Stoga možemo reći kako se drugo pravilo nalazi u „sjeni“ prvog pravila (engl. *shadowed*). Pri tome nije uopće važno koja je kombinacija akcija koje te dvije naredbe primjenjuju na prepoznati mrežni promet jer se akcija drugog pravila neće nikad realizirati. Formalno, pravilo R_y se nalazi u „sjeni“ pravila R_x ako jedan od sljedeća dva uvjeta vrijedi:

- R_x [pozicija] < R_y [pozicija]; $R_x \ \Re_{PP} \ R_y$; R_x [akcija] $\neq R_y$ [akcija]
- R_x [pozicija] < R_y [pozicija]; $R_y \ \Re_{UP} \ R_x$; R_x [akcija] $\neq R_y$ [akcija]



Slika 4: Originalna anomalija zasjenjenja

Takva definicija vrijedi u uvjetima kada postoje samo dva tipa akcija: prihvaćanje ili odbacivanje mrežnog paketa. Ali ovaj rad obuhvaća širi aspekt mogućih akcija – ne samo akcije prihvaćanja ili odbacivanja već i akcije logiranja, preusmjeravanja paketa, mijenjanja podataka u paketima i sl. Zbog toga je ovako definiranu anomaliju potrebno proširiti.

Prvo, definira se anomalija u „sjeni iza“ (engl. *shadowed after*) koja predstavlja takvu kombinaciju sigurnosnih pravila gdje prvo pravilo (u definiranom poretku) označava sav mrežni promet koji označava i drugo pravilo, tj. mrežni promet koji označava drugo pravilo podskup je mrežnog prometa označenog prvim pravilom. U tom slučaju se može reći da se drugo pravilo nalazi u „sjeni iza“ prvog pravila. Akcija prvog pravila mora biti ili prekidajuća (procesuiranje mrežnog paketa u tom lancu naredbi se prekida) ili pravila moraju imati iste akcije i akcijske parametre. Između pravila se ne smiju pronaći druga pravila s neprekidajućom akcijom i akcijskim parametrima jer postoji mogućnost da takva pravila modificiraju paket koji potom može biti procesuiran preko „zasjenjenog“ pravila te se stoga to pravilo ne smije uklanjati. Formalno, pravilo R_y se nalazi u „sjeni iza“ pravila R_x ako jedan od sljedeća dva uvjeta vrijedi:

- R_x[pozicija] < R_y[pozicija]; R_y {$\Re_{UP}$, $\Re_{PP}$} R_x;
  R_x[akcija] $\in$ {prekidajuće akcije};
  $\nexists R_z$: R_x[pozicija] < R_z[pozicija] < R_y[pozicija];
  R_z[akcija] $\in$ {neprekidajuće akcije}; $\exists R_z$[j]: j$\in$ {akcijski parametri}
- R_x[pozicija] < R_y[pozicija]; R_y {$\Re_{UP}$, $\Re_{PP}$} R_x; R_x[akcija] = R_y[akcija];
  $\forall$ i: R_x[i] = R_y[i], i $\in$ {akcijski parametri};
  $\nexists R_z$: R_x[pozicija] < R_z[pozicija] < R_y[pozicija];
  R_z[akcija] $\in$ {neprekidajuće akcije}; $\exists R_z$[j]: j$\in$ {akcijski parametri}

Druga modifikacija anomalije zasjenjenja je anomalija u „sjeni ispred“ (engl. *shadowed before*). koja predstavlja kombinaciju sigurnosnih pravila u kojoj drugo pravilo (u definiranom poretku) označava sav mrežni promet koji

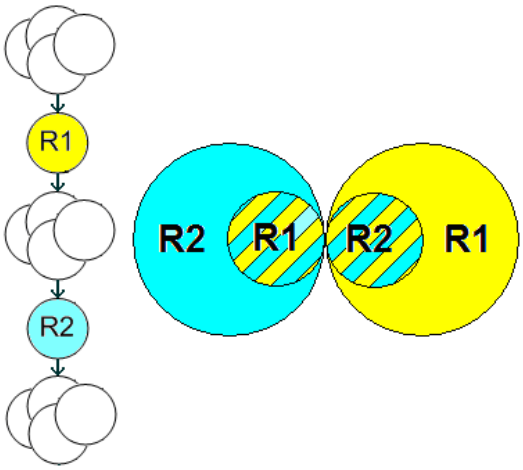
je označen i prvim pravilom, tj. mrežni promet koji označava prvo pravilo podskup je mrežnog prometa označenog drugim pravilom. Pri tome je bitno da akcije tih pravila budu identične (kao i eventualni akcijski parametri) te da se između njih ne smiju pronaći druga pravila s kojima se oni nalaze u relaciji, a da imaju drugačiju akciju i/ili akcijske parametre. Dodatno, ako prvo pravilo ima neprekidajuću akciju i barem jedan akcijski parametar između njih se ne smiju nalaziti nikakva pravila koja imaju neprekidajuću akciju i akcijske parametre. Ali ako dva promatrana pravila imaju neprekidajuću akciju i nemaju akcijskih parametara tada se između njih smiju pronaći pravila s kojima se oni nalaze u relaciji i koja imaju različitu neprekidajuću akciju te nemaju akcijske parametre. Formalno, pravilo R_x se nalazi u „sjeni ispred“ pravila R_y ako jedan od sljedeća tri uvjeta vrijedi:

- R_x[pozicija] < R_y[pozicija]; R_y {$\Re_{UP}$, $\Re_{PP}$} R_x; R_x[akcija] = R_y[akcija];
  $\forall$ i: R_x[i] = R_y[i], i $\in$ {akcijski parametri};
  $\nexists$ (R_x[akcija] $\in$ {neprekidajuće akcije}; $\exists$ R_x[j]: j$\in$ {akcijski parametri})
  $\nexists$ R_z: R_x[pozicija] < R_z[pozicija] < R_y[pozicija]; R_x {$\Re_R$} R_z;
  R_x[akcija] $\neq$ R_z[akcija];
  $\nexists$ R_w: R_x[pozicija] < R_w[pozicija] < R_y[pozicija]; R_x {$\Re_R$} R_w;
  $\exists$ k: R_x[k] $\neq$ R_w[k] ,k$\in$ {akcijski parametri}
- R_x[pozicija] < R_y[pozicija]; R_y {$\Re_{UP}$, $\Re_{PP}$} R_x; R_x[akcija] = R_y[akcija];
  R_x[akcija] $\in$ {neprekidajuće akcije}; $\exists$ R_x[j]: j $\in$ {akcijski parametri};
  $\forall$ i: R_x[i] = R_y[i], i $\in$ {akcijski parametri};
  $\nexists$ R_z: R_x[pozicija] < R_z[pozicija] < R_y[pozicija]; R_x {$\Re_R$} R_z;
  R_x[akcija] $\neq$ R_z[akcija];
  $\nexists$ R_w: R_x[pozicija] < R_w[pozicija] < R_y[pozicija]; R_x {$\Re_R$} R_w;
  $\exists$ k: R_x[k] $\neq$ R_w[k] ,k$\in$ {akcijski parametri}
  $\nexists$ R_q: R_x[pozicija] < R_q[pozicija] < R_y[pozicija];
  R_q[akcija] $\in$ {prekidajuće akcije}
  $\nexists$ R_o: R_x[pozicija] < R_o[pozicija] < R_y[pozicija];
  R_o[akcija] $\in$ {neprekidajuće akcije}; $\exists$ R_o[p]: p$\in$ {akcijski parametri}
- R_x[pozicija] < R_y[pozicija]; R_y {$\Re_{UP}$, $\Re_{PP}$} R_x; R_x[akcija] = R_y[akcija];
  R_x[akcija] $\in$ {neprekidajuće akcije};
  $\nexists$ R_x[j]: j $\in$ {akcijski parametri}; $\nexists$ R_y[i], i $\in$ {akcijski parametri};
  $\nexists$ R_z: R_x[pozicija] < R_z[pozicija] < R_y[pozicija]; R_x {$\Re_R$} R_z;
  R_z[akcija] $\in$ {prekidajuće akcije};
  $\nexists$ R_w: R_x[pozicija] < R_w[pozicija] < R_y[pozicija]; R_x {$\Re_R$} R_w;
  $\exists$ R_w[k] ,k$\in$ {akcijski parametri}

Drugi uvjet definira da između promatranih pravila ne smiju postojati pravila koja imaju akcijske parametre. Takva pravila mogla bi promijeniti mrežni paket te ga staviti u obuhvat ili ga maknuti iz obuhvata drugog pravila. Nažalost, Firo trenutačno ne klasificira akcijske parametre, ali novije verzije bi trebale povezati akcijske parametre s parametrima uspoređivanja. Kada bi među-pravila imala akcijske parametre koji bi mijenjali parametre paketa, konekcije ili drugih elemenata koje promatrano pravilo ne koristi za uspoređivanje, tada bi bilo dozvoljeno uklanjanje pravila koje se nalazi u „sjeni ispred“.

3.2. Anomalija korelacije

Anomalija korelacije predstavlja takvu kombinaciju sigurnosnih pravila u kojoj prvo pravilo označava dio mrežnog prometa koji označava i drugo pravilo, ali pri tome ta dva pravila imaju različite akcije tj. jedno pravilo prihvaća mrežni promet dok ga drugo odbacuje. U takvoj situaciji je bitno koje pravilo se nalazi prvo u poretku jer će ono odlučivati da li će se mrežni promet koji oba pravila označavaju, odbaciti ili prihvatiti.



Slika 5: Anomalija korelacije

Formalno, pravilo R_y se nalazi *u korelaciji* s pravilom R_x ako vrijedi sljedeći uvjet:

- $R_y \Re_C R_x$; R_x [akcija] $\neq R_y$ [akcija]

Dvojbeno je treba li ovu anomaliju korelacije smatrati anomalijom ili ne. Naime, često je potrebno napisati dva različita pravila tako da ona označavaju isti mrežni promet. To je nužno jer bi u suprotnom trebalo napisati veći broj sigurnosnih pravila. Kao primjer mogu poslužiti sljedeća dva jednostavna pravila:

Tablica 7: Primjer sigurnosnih pravila u korelaciji

Pravilo	Protokol	Izvorna adresa	Ciljna adresa	Akcija
R1	UDP	10.0.0.0/16	10.1.4.0/24	Odbaci
R2	ALL	10.0.5.0/24	10.1.0.0/16	Prihvati

Navedena pravila R2 i R3 nalaze se u korelaciji. Mrežni promet koji oba pravila označavaju je UDP promet koji dolazi iz mreže 10.0.5.0/24 te ide na mrežu 10.1.4.0/24. Kako bi se kreirala pravila kod kojih se ne pojavljuje anomalija korelacije bilo bi potrebno kreirati veći broj pravila:

Tablica 8: Primjer sigurnosnih pravila nakon uklanjanja anomalije korelacije

Pravilo	Protokol	Izvorna adresa	Ciljna adresa	Akcija
R1	UDP	10.0.0.0/16	10.1.4.0/24	Odbaci
R2-1	ALL	10.0.5.0/24	10.1.0.0/22 (10.1.0.0 - 10.1.3.255)	Prihvati
R2-2	ALL	10.0.5.0/24	(10.1.5.0 - 10.1.255.255)	Prihvati

U navedenom primjeru drugo pravilo R2 je razdijeljeno na dva nova pravila R2-1 i R2-2. Veći broj sigurnosnih pravila narušava performanse vatrozida za sve mrežne pakete koji umjesto da se uspoređuju samo s pravilom R2, uspoređuju se sa pravilima R2-1 i R2-2.

Stoga kod anomalija korelacije nije preporučljivo primjenjivati automatske metode promjene sigurnosnih pravila. Eventualno može biti korisno administratora upozoriti na nastalu situaciju kako bi sam mogao prosuditi da li je kod definiranja sigurnosnih pravila napravio pogrešku.

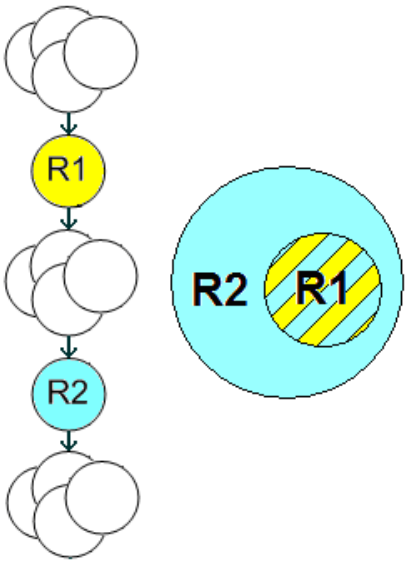
Budući da Firo uključuje u svom radu složene vrijednosti (inverzije, IP *wildcard* maske, nizove), broj pravila koji bi se izbjegavanjem anomalija korelacija iz jednog pravila generirao u nekim slučajevima mogao bi doseći nekoliko desetaka ili stotina pravila. Stoga ova anomalija nije dio Firo optimizatora niti se generiraju upozorenja.

3.3. Anomalija generalizacije

Anomalija generalizacije predstavlja takvu kombinaciju sigurnosnih pravila da sav mrežni promet koji je označen pomoću prvog sigurnosnog pravila označen u potpunosti i drugim pravilom, ali pri tome ta dva pravila imaju različite akcije, tj. jedno pravilo prihvaća mrežni promet dok ga drugo odbacuje. Anomalija generalizacije specijalni je slučaj prethodno opisane anomalije korelacije pri čemu prvo pravilo označava izuzetak, posebni slučaj u odnosu na drugo pravilo. Stoga vjerujem da tu vrijedi praksa administratora koji definiraju jedno pravilo sa suprotnom akcijom prije pravila koje obuhvaća veći spektar mrežnog prometa kako bi minimalizirali broj sigurnosnih pravila.

Formalno, pravilo R_y je *generalizacija* pravila R_x ako vrijedi sljedeći uvjet:

- R_x [pozicija] $\neq R_y$ [pozicija]; $R_x \ \Re_{UP} \ R_y$; R_x [akcija] $\neq R_y$ [akcija]



Slika 6: Anomalija generalizacije

Kao i kod anomalije korelacije, tako i kod anomalije generalizacije nije preporučljivo primjenjivati automatske metode promjene sigurnosnih pravila već eventualno administratora upozoriti na nastalu situaciju kako bi sam mogao prosuditi da li je kod definiranja sigurnosnih pravila namjerno kreirao pravilo generalizacije.

Budući da Firo ne radi samo s akcijama prihvaćanja i odbacivanja već omogućuje i optimiranje pravila s drugim oblicima akcija i akcijskih pravila, ova anomalija se ne prijavljuje i ne uklanja.

3.4. Anomalija redundantnosti

Anomalija redundantnosti označava ona sigurnosna pravila koja su redundantna, tj. postoji drugo sigurnosno pravilo koje označava iste mrežne pakete te ima istu akciju kao i redundantno pravilo pa ukoliko bi redundantno pravilo bilo uklonjeno to ne bi imalo nikakvog efekta na ukupnu sigurnosnu politiku. U grupu anomalija redundantnosti spadaju i prethodno opisane anomalije u „sjeni iza“ i u „sjeni ispred“.

Sva redundantna pravila predstavljaju pogrešku u definiranoj sigurnosnoj politici. Redundantno pravilo ne pridonosi filtriranju već naprotiv povećava veličinu tablice pravila za filtriranje te može povećati vrijeme pretraživanja pravila i nepotrebno koristiti resurse.

U sklopu ovog rada definirane su dvije nove anomalije redundantnosti koje nisam primijetio u radovima drugih autora, a to su redundantnost parametara u pravilima i redundantnost elemenata u parametrima.

Redundantnost parametara postoji kada pravilo ima određeni parametar uspoređivanja koji zahtijeva dodatno procesuiranje, iako je postojanje takvog parametara nepotrebno zbog drugih sigurnosnih pravila koja prethode analiziranom pravilu. Naime, parametar u analiziranom pravilu zajedno s parametrom iz prethodnog pravila pokriva sve moguće vrijednosti. Dakle, parametar iz drugog pravila je moguće ukloniti. Preduvjet za uklanjanje parametra iz drugog pravila je akcija prvog pravila koja mora biti prekidajuća te svi ostali parametri uspoređivanja moraju biti identični. Akcija drugog pravila može biti bilo kojeg tipa. Također, prije drugog pravila mogu se pronaći i neka ostala pravila s kojima se ono nalazi u relaciji, ali ta među-pravila moraju biti ili prekidajuća ili neprekidajuća bez akcijskih parametara koji bi mogli izmijeniti paket, konekciju ili druge podatke koje pravila provjeravaju.

Formalno, parametar *a* pravila R_y je redundantan ako vrijedi sljedeći uvjet:

- $\exists\ R_x$: R_x[pozicija] < R_y[pozicija]; R_x[akcija] $\in$ {prekidajuće akcije};
  $\forall$ i: R_x[i] = R_y[i], i $\in$ { {parametri uspoređivanja} \ { *a* } };
  R_x[*a*] $\cup$ R_y[*a*] = { sve moguće vrijednosti };
  $\nexists R_z$: R_x[pozicija] < R_z[pozicija] < R_y[pozicija];
  R_z[akcija] $\in$ {neprekidajuće akcije}; $\exists R_z$[j]: j$\in$ {akcijski parametri}

Važno je istaknuti kako akcija prvog pravila mora biti prekidajuća. Jer ukoliko bi se ova optimizacijska tehnika primijenila na dva neprekidajuća pravila s identičnom akcijom, tada bi npr. ukoliko se radi o logiranju isti paket bio dva puta logiran jer bi iz drugog pravila bio maknut parametar koji ograničava broj paketa na koji će se primijeniti. Stoga prva akcija mora biti prekidajuća.

Redundantnost elementa u parametru pravila postoji ako prije tog pravila postoji drugo pravilo koje već sadrži taj parametar i taj element parametra pri čemu su svi ostali parametri uspoređivanja u pravilima identični. Pravilo koje prethodi mora imati prekidajuću akciju ili mu akcija i akcijski parametri moraju biti identični kao kod drugog pravila. Također, između pravila se ne smije nalaziti pravilo koje ima neprekidajuću akciju i barem jedan akcijski parametar.

Formalno, element *E* parametra *a* u pravilu R_y je redundantan ako vrijedi jedan od sljedeća dva uvjeta:

- $\exists\ R_x$: R_x[pozicija] < R_y[pozicija]; R_x[akcija] $\in$ {prekidajuće akcije};
  $\forall$ i: R_x[i] = R_y[i], i $\in$ { {parametri uspoređivanja} \ { *a* } };
  E $\in$ R_x[*a*]; *E* $\in$ R_y[*a*];
  $\nexists R_z$: R_x[pozicija] < R_z[pozicija] < R_y[pozicija];
  R_z[akcija] $\in$ {neprekidajuće akcije}; $\exists R_z$[j]: j$\in$ {akcijski parametri}

- $\exists\ R_x$: R_x[pozicija] < R_y[pozicija]; R_x[akcija] = R_y[akcija];
  $\forall$ k: R_x[k] = R_y[k], k $\in$ {akcijski parametri};
  $\forall$ i: R_x[i] = R_y[i], i $\in$ { {parametri uspoređivanja} \ { *a* } };
  E $\in$ R_x[*a*]; *E* $\in$ R_y[*a*];

$\nexists R_z$: R_x[pozicija] < R_z[pozicija] < R_y[pozicija];
R_z[akcija] $\in$ {neprekidajuće akcije}; $\exists R_z$[j]: j$\in$ {akcijski parametri}

Uklanjanje redundantnih elemenata iz parametara trenutno je usmjereno samo na uklanjanje portova iz niza portova, kao i na uklanjanje elemenata iz niza znakovnih vrijednosti. Uklanjanje se obavlja jer ti elementi već postoje u nekom prethodnom pravilu. Time pravila postaju manje složena, a ponekad se dobiveni niz portova može konvertirati u jednostavniji raspon ili se izmjenom pravila ono može koristiti u drugim optimizacijskim metodama.

Iako kod Iptables alata nije moguće imati neprekidajuću akciju koja u akcijskim parametrima mijenja portove ili niz znakovnih vrijednosti, ipak je u prethodnim uvjetima radi općenitosti zadržan uvjet po kojem se između promatranih pravila ne smije nalaziti pravilo koje ima neprekidajuću akciju i barem jedan akciji parametar.

3.5. Anomalija irelevantnosti

U svojim kasnijim radovima [8], E. S. Al-Shaer i H. H. Hamed definirali su anomaliju irelevantnosti koja označava sva ona sigurnosna pravila koja označavaju mrežni promet koji se nikad neće pojaviti na vatrozidu. Po originalnoj definiciji tu se misli na izvorne i ciljne IP adrese. Rješavanje ovog pravila uglavnom iziskuje poznavanje uvjeta u kojima vatrozid radi. Prvenstveno je potrebno znati koje su moguće izvorne i ciljne IP adrese. Ukoliko se u parametrima sigurnosnih pravila nalaze one IP adrese koje se ne nalaze u skupu mogućih izvornih i ciljnih IP adresa tada je ta sigurnosna pravila moguće ukloniti.

Za uklanjanje ovih anomalija nije dovoljna samo analiza postojeće liste sigurnosnih pravila već je potrebno analizirati i mreže na koje je vatrozid spojen. Najčešće vatrozid ima definiranu rutu za mrežu 0.0.0.0/0 koja označava sve moguće IP adrese i primjenjuje se kao predefinirana ruta (engl. *default gateway*) te stoga tu ne postoji mogućnost detektiranja anomalija irelevantnosti baziranih na IP adresama.

Ali „nemogući“ irelevantni mrežni paketi ne moraju biti definirani samo po IP adresama, već mogu biti definirani i po dozvoljenim protokolima, portovima, TCP parametrima i slično; ali za poznavanje takvih uvjeta potrebni su složeniji programi.

3.6. Uzroci i učestalost anomalija

Vjerojatnost pojave anomalija raste sa složenošću potreba organizacije. Što su one složenije to će definirane vatrozidne politike biti duže i s većim brojem kompleksnijih sigurnosnih pravila. A nije iznenađujuće što postoji snažna korelacija između složenosti vatrozida i broja pogrešaka u njima. Do

problema uobičajeno dolazi kad se sigurnosne politike mijenjaju, npr. određeni odjeli dobiju nove ovlasti, a ne postoje efikasne politike upravljanja promjenama, niti dobro definirani poslovni zahtjevi.

Postoji jedna izreka glede upravljanja vatrozidima – „Što ulazi unutra, a nikad ne izlazi van?“. Odgovor na to pitanje je – sigurnosno pravilo. Naime, većina organizacija ima dobro definirane politike za dodavanje novih pravila na vatrozid, ali ne i za njihovo uklanjanje. Secure Passage [55] je u sklopu svoje analize identificirao kako je u 63% analiziranih slučajeva neuklonjena pravila bila glavni uzrok kompleksnosti. Njihov alat FireMon prati takva sigurnosna pravila te identificira sigurnosna pravila koja su neiskorištena, tj. niti jedan mrežni paket nije na njima uspješno prepoznat.

Autori koji su definirali anomalije proveli su i analizu njihove učestalosti [9] na način da su dvanaestorici administratora s različitim razinama iskustva dali kao zadatak kreiranje sigurnosnih pravila za definiranu sigurnosnu politiku. Administratori su u prosjeku unosili 40 pravila koje su potom autori analizirali te detektirali učestalost pojave anomalija. Rezultati su vidljivi u sljedećoj tablici:

Tablica 9: Prosjek otkrivenih anomalija u sigurnosnim pravilima podijeljeni po razini stručnosti administratora

Stručnost administratora	Zasjenjenje	Redundantnost	Korelacija	Irelevantnost
Stručnjak	0%	5%	3%	0%
Prosječan	1%	9%	3%	0%
Početnik	4%	12%	9%	2%

Izvor: Al-Shaer E. S., Hamed H. H. (2004): Modeling and Management of Firewall Policies, IEEE Transactions Network and Service Management, vol. 1. no 1; str 2-10.

Iz podataka je vidljivo kako su najmanje griješili iskusni administratori (8%), ali su čak i oni činili pogreške. Najviše su griješili početnici (27%), od kojih je to bilo i za očekivati.

Uz sve prethodno navedene razloge, očito je kako velik dio problema može riješiti iskusan administrator, ali čak ni on ne može izbjeći sve anomalije. A kada je broj sigurnosnih pravila velik, za razliku od navedenog eksperimenta (samo 40 pravila), može se očekivati i veći broj anomalija.

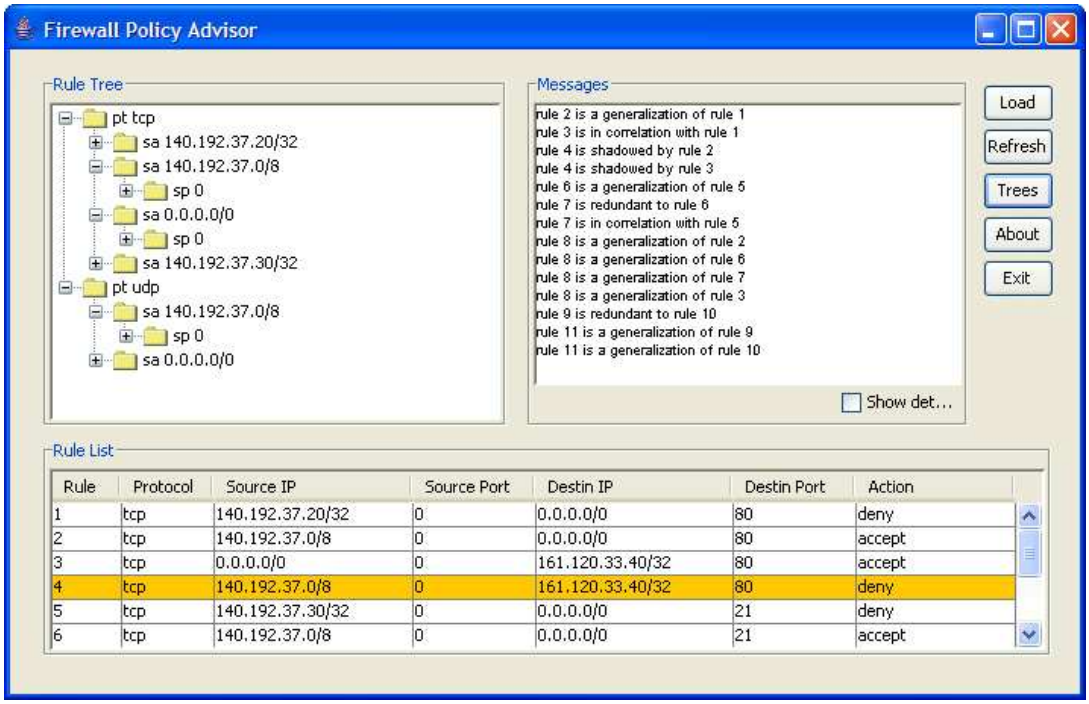
4. Metode optimiranja poretka sigurnosnih pravila

Problematikom optimiranja sigurnosnih pravila računalni stručnjaci počeli su se baviti tek početkom 21. stoljeća. Glavne metode optimiranja sigurnosnih politika mogu se podijeliti na statičke i dinamičke. **Statičke** metode prilikom optimiranja promatraju samo niz sigurnosnih politika definiranih na nekom vatrozidu; dok **dinamičke** metode uključuju i druge informacije kao što je vjerojatnost i mogućnost pojave mrežnih paketa. Za dinamičke metode ključno je analiziranje mrežnog prometa u prošlosti kako bi se mogle donositi odluke u budućnosti dok to kod statičkih metoda nije potrebno. Cohen i Lund [21] pokazali su kako mrežni promet obično prati Zipf distribuciju pri čemu većinu mrežnog prometa obrađuje manjina pravila, a cilj dinamičkih metoda optimiranja je identificirati ta pravila.

Druga podjela metoda optimiranja je na metode koje procesuiraju **jedinstvenu** sigurnosnu politiku – sigurnosna pravila definirana samo na jednom vatrozidu, te na metode koje procesuiraju **distribuirane** sigurnosne politike - sigurnosna pravila na većem broju vatrozida.

4.1. Statičko optimiranje korištenjem stabla naredbi

Ehab S. Al-Shaer i Hazem H. Hamed, koji su postavili klasifikaciju anomalija, razvili su alat pod nazivom Firewall Policy Advisor [6], [7] za predlaganje rješavanja anomalija u sigurnosnim pravilima vatrozida.



Slika 7: Firewall Policy Advisor alat za upozoravanje na anomalije

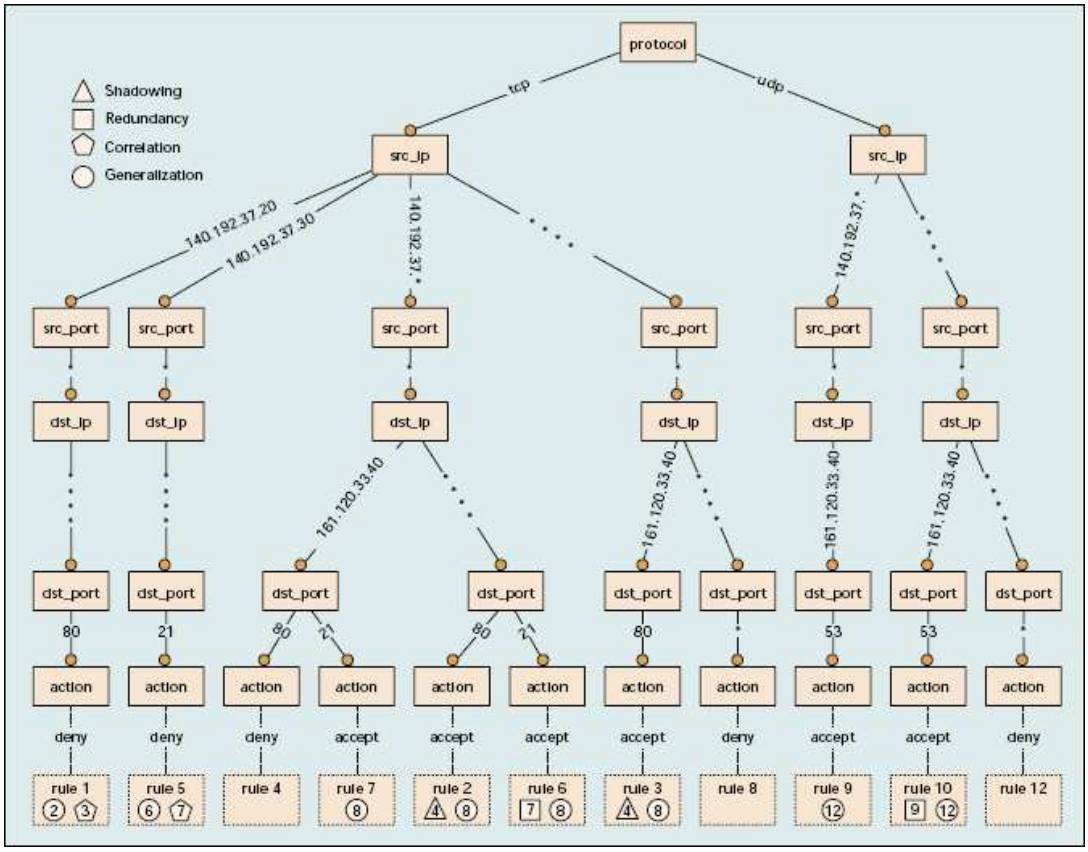
Alat je baziran na izgradnji stabla sigurnosnih naredbi kroz koje se potom pretražuju sljedeće anomalije: zasjenjenje, korelacija, generalizacija i redundantnost. Na sljedećoj slici Slika 8 dano je 12 sigurnosnih pravila za koje je potom izgrađeno stablo naredbi koje je prikazano na slici
Slika **9**.

	Izvorište		Odredište		
Protokol	Adresa	Port	Adresa	Port	Akcija
1: tcp,	140.192.37.20,	svi,	*.*.*.*,	80,	odbaci
2: tcp,	140.192.37.*,	svi,	*.*.*.*,	80,	prihvati
3: tcp,	*.*.*.*,	svi,	161.120.33.40,	80,	prihvati
4: tcp,	140.192.37.*,	svi,	161.120.33.40,	80,	odbaci
5: tcp,	140.192.37.30,	svi,	*.*.*.*,	21,	odbaci
6: tcp,	140.192.37.*,	svi,	*.*.*.*,	21,	prihvati
7: tcp,	140.192.37.*,	svi,	161.120.33.40,	21,	prihvati
8: tcp,	*.*.*.*,	svi,	*.*.*.*,	svi,	odbaci
9: udp,	140.192.37.*,	svi,	161.120.33.40,	53,	prihvati
10: udp,	*.*.*.*,	svi,	161.120.33.40,	53,	prihvati
11: udp,	140.192.38.*,	svi,	161.120.35.*,	svi,	prihvati
12: udp,	*.*.*.*,	svi,	*.*.*.*,	svi,	odbaci

Slika 8: Primjer sigurnosnih naredbi (Al-Shaer E. S., Hamed H. H., 2004)

Prilikom izgradnje stabla sigurnosnih pravila koja jednim dijelom podsjećaju na binarne dijagrame odluke (engl. BDD – *Binary Decision Diagrams*) [34], odabire se jedan parametar koji predstavlja čvor pri čemu moguće vrijednosti tog čvora čine grane stabla.

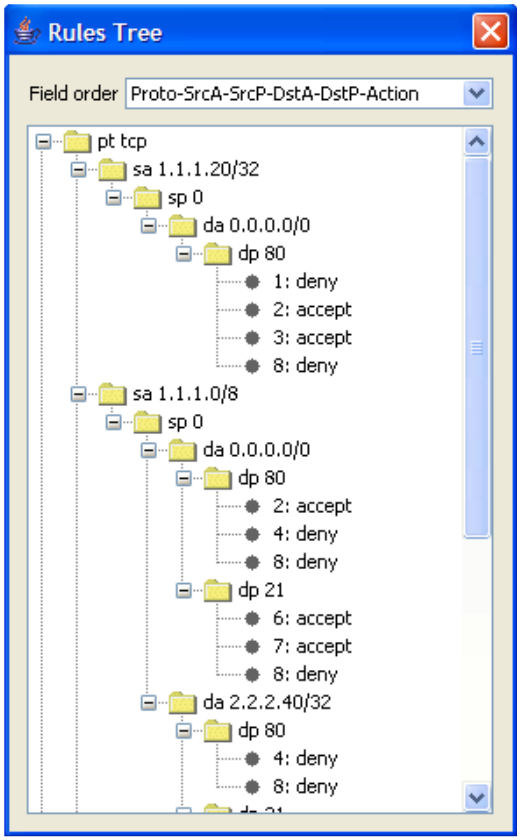
U startu između sigurnosnih pravila ne postoje nikakve relacije. Svaki parametar pravila R_y se uspoređuje s pripadnim parametrom u pravilu R_x počevši s protokolom pa potom izvornom IP adresom i portom te konačno ciljnom IP adresom i portom. Pretpostavka je da se pravilo R_y nalazi iza R_x. Odnos ispitivanih pravila ovisi o rezultatima usporedbe. U slučaju kad je svaki parametar R_y podskup ili jednak ekvivalentnom parametru naredbe R_x, a oba pravila imaju istu akciju, radi se o redundanciji pravila R_y u odnosu na pravilo R_x. U slučaju kad imaju suprotne akcije radi se o anomaliji zasjenjenja. U slučaju kad imaju suprotne akcije, a R_x se nalazi iza R_y, R_y predstavlja generalizaciju pravila R_x.



Slika 9: Stablo sigurnosnih naredbi (Al-Shaer E. S., Hamed H. H., 2004)

Korelacija između pravila postoji kad su neki parametri R_y podskupovi ili jednaki ekvivalentnim parametrima naredbe R_x, a istovremeno su neki parametri R_x podskupovi ili jednaki ekvivalentnim parametrima naredbe R_y.

Tijekom rada, nova pravila se dodaju u stablo i uspoređuju s postojećima. U slučaju postojanja anomalije, generira se izvještaj zajedno sa sigurnosnim naredbama koje su je uzrokovale.



Slika 10: Firewall Policy Advisor – vizualizacija stabla sigurnosnih naredbi

4.2. Ostale statičke metode optimiranja

Pasi Eronen i Jukka Zitting [25] su još 2001. prije prvih pojava definicija anomalija u vatrozidima predložili ekspertni sustav koji je zasnovan na CLPu (engl. *Contraint Logic Programming*) što omogućava korisnicima samostalno definiranje učestalih konfiguracijskih pogrešaka. Baza znanja je skup pravila i činjenica o sigurnosnim pravilima izraženih u Prolog jeziku. Time se mogu definirati dozvoljene vrijednosti protokola, izvornih i ciljnih IP adresa i portova, karakteristike TCP, UDP konekcija i dr., a potom na temelju tih podataka i na temelju definirane liste sigurnosnih pravila identificirati administratorske konfiguracijske pogreške.

Thawatchai Chomsiri i Chotipat Pornavalai [20] predložili su svoju metodu analiziranja i optimiranja sigurnosnih pravila vatrozida zasnovanu na relacijskoj algebri i tzv. Raining 2D-Box modelu. Novost koju su uveli time je da se prilikom optimiranja ne uspoređuju samo 2 sigurnosna pravila, već se usporedbe obavljaju na većem broju pravila.

Na temelju rada [22] o razdjeljivanju sigurnosnih pravila kako ona ne bi bila u relaciji tj. kako ne bi identificirali isti mrežni promet, definirana je metoda [3] statičkog optimiranja kod koje se niz sigurnosnih pravila prvo razdjeljuje na veći broj pravila (komponenta DSC – *Disjoint Set Creator*)

nakon čega se ta pravila spajaju (komponenta DSM – *Disjoint Set Merger*) uz izbjegavanje preklapanja među pravilima.

Mohsen Rezvani i Ramtin Aryan [52] uključili su u proces optimiranja i formalni jezik za predstavljanje sigurnosnih pravila na temelju kojeg se definirana sigurnosna politika, jednostavne anomalije (između dva sigurnosna pravila) te složene anomalije (između dvije grupe pravila) prevode u logičke formule. Korištenjem formalnog modela i binarnih dijagrama odlučivanja anomalije se detektiraju i uklanjaju.

Postoji više radova koji analiziraju postojeće ili daju svoje algoritme za pronalaženje i uklanjanje anomalija [23], [19], [17], [32], [44], [2]. Čak su analizirane i brzine optimiranja sigurnosnih pravila [46]. Ipak, kad se promatraju statički optimizatori, ni brzine ni metode nisu pretjerano bitne jer ti se optimizatori pokreću samo jednom i nije važno traju li milisekundu ili čak minutu. Zaključno, najbolji algoritam nije moguće odrediti i ne postoji najbolji za svaki scenarij [43]. Kod svih analiziranih statičkih optimizatora jedino su bili važni konačni rezultati – što je optimirano.

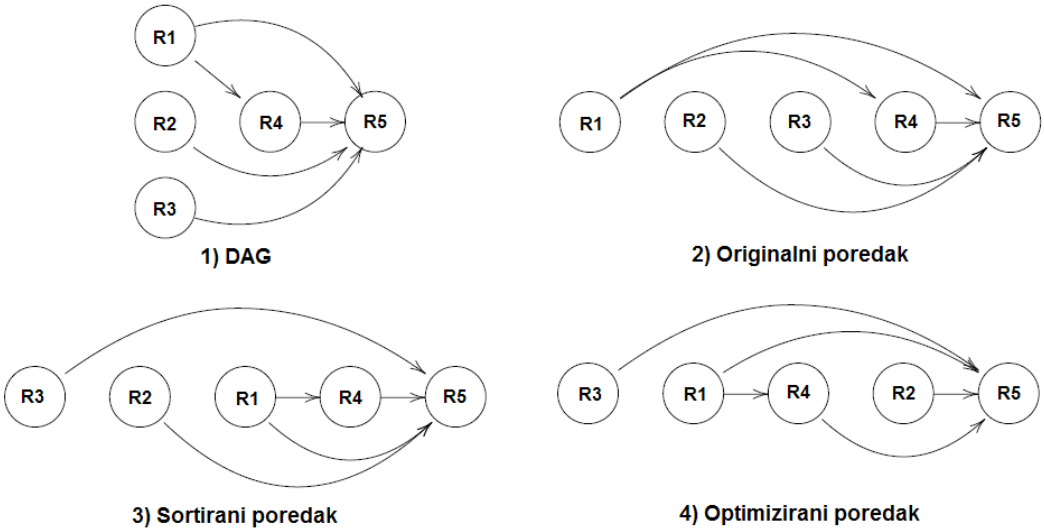
4.3. Dinamičko optimiranje pomoću direktnih acikličkih grafova

Princip direktnih acikličkih grafova [27], [56] (engl. DAG – *Directed Acyclic Graphs*) spada u dinamičke metode optimiranja, a najjednostavnije ga je objasniti preko primjera pa je u stoga u nastavku dan primjer pet sigurnosnih pravila s vjerojatnošću (učestalosti) zadovoljavanja tih sigurnosnih pravila (npr. TCP mrežni promet koji ima izvorne i ciljne IP adrese iz 2.0.0.0/8 mreže te cilljni port 20 ima vjerojatnost 0.25 tj. ono čini 25% svog mrežnog prometa). Te vjerojatnosti dobiju se prethodnom analizom mrežnog prometa kroz neki vremenski period.

Tablica 10: Primjer vatrozida od pet sigurnosnih pravila

	Protokol	Izvorni IP	Izvorni port	Ciljni IP	Ciljni port	Akcija	Vjerojatnost
R1	UDP	1.1.*	svi	Svi	80	odbaci	0.01
R2	TCP	1.*	svi	1.*	90	prihvati	0.02
R3	TCP	2.*	svi	2.*	20	prihvati	0.25
R4	UDP	1.*	svi	Svi	Svi	prihvati	0.22
R5	svi	svi	svi	Svi	Svi	odbaci	0.5

Na temelju podudarnosti parametara sigurnosnih pravila odrede se zavisnosti između pojedinih pravila. Budući da je važno sačuvati poredak, zavisnosti se označavaju strelicama kao što je prikazano na sljedećoj slici.



Slika 11: Koraci DAG optimizacije

Prethodna slika prikazuje četiri koraka prilikom optimiranja sigurnosnih pravila. U prvom koraku se odredi na koji način su sigurnosna pravila u zavisnosti pri čemu strelice postoje između onih pravila koje se nalaze u relaciji, a strelica je okrenuta prema onom pravilu koje se nalazi kasnije u listi. U drugom koraku se ta zavisnost prikaže, ali nad originalnim poretkom sigurnosnih pravila. U trećem koraku se sigurnosna pravila sortiraju prema vjerojatnosti pravila kojima ne prethode druga pravila (ne postoji strelica koja u njih pokazuje – R1, R2 i R3) pri čemu se pravila s većom vjerojatnošću smještaju bliže početka niza. Stoga je R3 s vjerojatnošću 0.25 na prvom mjestu, zatim R2 s vjerojatnošću 0.02 te potom R1 s vjerojatnošću 0.01. U zadnjem koraku se za potrebe optimizacije ne promatraju samo pojedinačna sigurnosna pravila već se ona grupiraju sa svojim zavisnim pravilima. Zato se kombinacija pravila R1 i R4 s ukupnom vjerojatnošću 0.23 (0.1 + 0.22) stavlja ispred pravila R2 koje ima vjerojatnost samo 0.02.

Iako ovaj rad ne uključuje dinamičko optimiranje i vjerojatnosti pojave mrežnog prometa način rada Firo optimizatora može se prikazati modificiranim DAG konekcijama, budući da se između svih pravila koji se nalaze u relacijama kreiraju konekcije. Svako pravilo ima listu konekcija prema pravilima s kojima se nalazi u relaciji. Postoje dvije liste, prva za pravila koja mu prethode, a druga za pravila koja mu slijede. Te konekcije koriste se prilikom odlučivanja smije li se koja redundancija ukloniti, a mijenjaju se prilikom uklanjanja pojedinih pravila.

4.4. Dinamičko optimiranje zasnovano na karakteristikama mrežnog prometa

Organizacije nemaju u isto doba dana, iste dane u tjednu, ili iste mjesece u godini istu strukturu mrežnog prometa. Osnovna ideja optimiranja

zasnovanog na karakteristikama mrežnog prometa jest predvidjeti kretanja mrežnog prometa za različite vremenske periode te sukladno tome kreirati zasebne liste sigurnosnih pravila. U tim listama, ona sigurnosna pravila koja označavaju najvjerojatniji oblik mrežnog prometa se pozicioniraju bliže početku liste.

Ehab S. Al-Shaer i Hazem H. Hamed predstavili su i napredniju dinamičku metodu optimiranja [10], [30] koja je bazirana na karakteristikama mrežnog prometa i kojoj je glavna ideja pozicionirati sigurnosna pravila za mrežni promet koji se češće pojavljuje na prve pozicije, a sigurnosna pravila za mrežni promet koji se rjeđe pojavljuje na kasnije pozicije, pri čemu se ne smije promijeniti ukupna sigurnosna politika vatrozida. Posebnost takvog optimiranja je u tome što se poredak sigurnosnih pravila mijenja kroz vrijeme i prilagođava promjenama karakteristika mrežnog prometa.

U svom radu autori su analizirali osobine Internet mrežnog prometa koje je provedeno na Sveučilištu Auckland i došli do zaključka kako 20% mrežnih tokova sadrže 10 i više paketa te prenose 70% mrežnog prometa, tj. većina Internet prometa sačinjena je od malih paketa učestale pojave. Također, 20% mrežnih tokova traju 5 sekundi i više te prenose 60% mrežnog prometa. I kako bi se ubrzalo filtriranje mrežnih paketa, nužno je u prvom redu smanjiti procesuiranje najučestalijih paketa.

Kao i kod prethodne dinamičke metode optimiranja, ključno je poznavanje važnosti pojedinih tipova mrežnog prometa. Kod prethodne optimizacijske metode koristile su se vjerojatnosti, dok se kod ove metode primjenjuje računanje „težina“ sigurnosnih pravila. „Težina“ u sebi ne sadržava samo frekvenciju zadovoljavanja pravila (što je povezano s vjerojatnošću pojave), već uključuje i aktualnost zadovoljenja pravila.

Frekvencija zadovoljavanja sigurnosnog pravila F_i u nekom vremenskom intervalu, može se izraziti kao omjer broja paketa f_i uspješno prepoznatih na pravilu R_i i ukupnog broja paketa prepoznatih na vatrozidu:

$$F_i = f_i \Big/ \left(\sum_{j=1}^{n} f_j \right) \tag{1}$$

Aktualnost zadovoljenja sigurnosnog pravila T_i u nekom vremenskom intervalu može se izraziti kao omjer vremena t_i kad je paket zadnji put uspješno prepoznat na pravilu R_i i vremena t_{zadnji} kad je kad je paket zadnji put uspješno prepoznat na bilo kojem pravilu vatrozida:

$$T_i = t_i \Big/ t_{zadnji} \tag{2}$$

Iz prethodne dvije vrijednosti dobije se „težina“ sigurnosnog pravila:

$$w_i = (1 - \rho) F_i + \rho T_i \quad (3)$$

Faktor aktualnosti ρ definira koliko se „težina“ pravila treba pouzdati na aktualnost sigurnosnog pravila, jer ona je osjetljiva na tip prometa koji može biti ravnomjeran ili neravnomjeran.

Optimirana lista pravila kreira se na temelju izračunatih „težina“ pravila pri čemu pravila s većom težinom trebaju biti više pozicionirana. Ali čak i kad se poredak sigurnosnih pravila promijeni nije moguće očekivati kako će tako dobiveni poredak biti konačan jer se mrežni promet stalno mijenja. Stoga se poredak sigurnosnih pravila mijenja korištenjem dvaju okidača:

- vremenski okidač – u određeno vrijeme očekuje se promjena mrežnog prometa;
- performansni okidač – kad se uoči pad u performansama prepoznavanja mrežnih paketa; pad se identificira na temelju devijacije aktualnog prosječnog broja usporedbi od optimalnog prosječnog broja usporedbi.

Ipak, ovako definirano optimiranje ima i svoju negativnu stranu, a to je utjecaj na performanse obrade, jer se cijelo vrijeme prilikom procesuiranja mrežnih paketa bilježe usporedbe (brojači) te se u slučaju „okidača“ rade promjene u sigurnosnim pravilima. Ipak, uz detaljna podešavanja ukupni rezultati performansa uspoređivanja mogu biti pozitivni.

4.5. Ostale dinamičke metode optimiranja

Dinamičke optimizacije sigurnosnih pravila vatrozida mogu biti zasnovane na tekućim analizama koji prate aktualne osobine mrežnog prometa i/ili na dubinskoj analizi postojećih podataka (engl. *data mining*) što zahtijeva korištenje dnevničkih zapisa (engl. *log*) o ostvarenom mrežnom prometu. Praćenje mrežnog prometa kakvo je opisano u prethodnom poglavlju složenije je pa su učestalije metode dinamičkog optimiranja zasnovane na analizi *log* zapisa. Podaci koji se iz dnevničkih ili tzv. log zapisa mogu dobiti su različiti. Npr. moguće je iz log zapisa otkriti koliko je pojedino sigurnosno pravilo identificiralo mrežnih paketa te koliki je udio tih mrežnih paketa u cjelokupnom mrežnom prometu. To je najjednostavniji oblik korištenja log zapisa.

Kompliciraniji postupci dubinske analize podataka mogu otkriti kako u sigurnosnim politikama postoje određena pravila za servise u mreži koji npr. nisu aktivni. Također, dubinska analiza podataka omogućava detektiranje anomalija irelevantnosti gdje se u pravilima koriste IP adrese koje se nikad ne pojavljuju u log zapisima. Stoga je preporučljivo uključiti i neke druge izvore osim samih log zapisa u dubinsku analizu podataka kao npr. podatke o otvorenim servisima i portovima u lokalnoj mreži.

Korištenjem dubinske analize podataka moguće je otkriti i određene oblike mrežnog prometa za koje ne postoje definirana sigurnosna pravila. Budući da takvi oblici mrežnog prometa mogu imati vrlo veliku vjerojatnost pojave, na njih se primjenjuju sva ona pravila koja prethode glavnom sigurnosnom pravilu koje takav neprepoznati mrežni promet ili odbacuje ili prihvaća. Kako bi se spriječilo nepotrebno provjeravanje svih sigurnosnih pravila za otkrivene učestale oblike mrežnog prometa kreiraju se sigurnosna pravila koja označavaju te oblike mrežnog prometa i na njih primjenjuju osnovnu politiku prihvaćanja ili odbacivanja. Pri tome optimizacijske metode koje su zasnovane na dubinskoj analizi podataka moraju u kalkulacije uzeti i efekt novog pravila na mrežni promet koji se razlikuje od mrežnog prometa označenog tim pravilom. Naime, ukoliko to pravilo npr. označava 30% ukupnog mrežnog prometa te je umetnuto na prvu poziciju u listi pravila, ostalih 70% mrežnog prometa nepotrebno se uspoređuje s tim umetnutim pravilom. Ali 30% mrežnih paketa više se ne uspoređuje s npr. ostalih 100 pravila koja su postojala prije promjene. Zbog navedenog, potrebno je dobro izračunati pozitivne i negativne efekte umetnutih pravila te na koje ih je pozicije najbolje postaviti.

Upravo ovakav način optimiranja koje uključuje dodavanje novih sigurnosnih pravila predložila je grupa autora predvođena Subrata Acharyaom [3], [5] pri čemu je ta shema optimizacije nazvana *default proxy.* Na temelju osnovne sigurnosne politike - zadnje definirano ili nedefinirano pravilo koje se primjenjuje na sve neprepoznate mrežne pakete (najčešće je to akcija – Odbaci); kao i na temelju učestalosti mrežnih paketa koji se najčešće pojavljuju (odbacuju) na tom pravilu, dodaju se nova pravila koja takav mrežni promet odbacuju u ranijim koracima usporedbe. Uz navedenu shemu autori su predložili i standardno praćenje povijesnih dnevničkih zapisa na temelju kojih se identificiraju najčešće prepoznata pravila (tzv. *hot rules*) koja se u skladu s tim premještaju. Također je opisana i metoda koja na temelju tekućeg mrežnog prometa radi trenutne izmjene definiranih sigurnosnih pravila vatrozida (tzv. *total re-ordering* shema). I na kraju opisuju i četvrtu shemu tzv. *online adaptation* koja kombinira prethodne dvije – prvo se na temelju dnevničkih zapisa identificiraju najčešća pravila te se generira „dugoročni“ profil koji se mijenja kad se ustanove velika odstupanja između „kratkoročnog“ tekućeg i „dugoročnog“ predviđenog mrežnog prometa.

Grupa autora predvođena Muhammad Abedinom [1] orijentirala se samo na analizu postojećih dnevničkih zapisa; u svom radu prvo na temelju dubinske analize dnevničkih zapisa generiraju svoj niz primitivnih sigurnosnih pravila koja su najučestalija (najfrekventnija) – tzv. MLF (engl. *Mining firewall Log using Frequency*) tehnika. Nakon toga se ta pravila mijenjaju (agregiraju) korištenjem FRR (engl. *Firewall Rule Regenerating*) algoritma koji koristi agregaciju te metode zasnovane na heuristici. Slične algoritme predstavili su i autori okupljeni oko Korosh Golnabia [28].

Od ostalih dinamičkih optimizatora potrebno je spomenuti i OPTWALL [4], prilagodljivi hijerarhijski optimizator vatrozida u koji je uz *on-line*

prilagođavanje ugrađena i zaštita od DoS (engl. *Denial of Service*) napada pa tako ima i svrhu IPS-a (engl. *Intrusion Preventyion System*).

4.6. Optimiranje distribuiranih sigurnosnih politika

E. Al-Shaer i H. Hamed [8] definirali su i anomalije koje se mogu pojaviti kod distribuiranih vatrozida. Iako distribuirani vatrozidi nisu predmet ovog rada u nastavku su opisane i te anomalije.

Anomalija **zasjenjenja** se ostvaruje kada prvi vatrozid blokira mrežni promet koji drugi vatrozid dozvoljava. Pri tome mrežni promet prvo dolazi na prvi vatrozid, a tek potom na drugi vatrozid.

Lažna (engl. *spuriousness)* anomalija se ostvaruje kada prvi u nizu vatrozid dozvoljava mrežni promet koji drugi u nizu vatrozid blokira.

Anomalija **redundantnosti** se ostvaruje ukoliko drugi u nizu vatrozid blokira mrežni promet koji je već blokirao vatrozid prvi u nizu.

Anomalija **korelacije** predstavlja takvu kombinaciju sigurnosnih pravila da pravilo na jednom vatrozidu označava dio mrežnog prometa koji označava i pravilo na drugom vatrozidu, a pri tome ta dva pravila mogu imati bilo koju kombinaciju akcija (njihove akcije mogu biti iste ili različite).

Identificiranje svih navedenih anomalija autori su uklopili i u svoj Firewall Policy Advisor alat. Također su analizirali njihovu učestalost eksperimentom [11] u kojem su dvanaestorici administratora s različitim razinama iskustva dali kao zadatak kreiranje sigurnosnih pravila za definiranu sigurnosnu politiku. Administratori su u prosjeku unosili 90 pravila na 3 vatrozida koje su potom autori analizirali te identificirali učestalost pojave anomalija. Rezultati su vidljivi u sljedećoj tablici.

Tablica 11: Prosjek otkrivenih anomalija u distribuiranim sigurnosnim pravilima podijeljeni po razini stručnosti administratora

Stručnost administratora	Zasjenjenje	Lažnost	Redundantnost	Korelacija
Stručnjak	1%	7%	9%	1%
Prosječan	3%	10%	11%	2%
Početnik	6%	14%	17%	2%

Izvor: Al-Shaer E. S., Hamed H. H., Boutaba R., Hasan M. (2005): Conflict classification and Analysis of Distributed Firewall policies, IEEE journal on selected areas in Communications, 23 (10), str. 2069-2084.

Usporedbom s tablicom Tablica 9 koja prikazuje učestalost pojave anomalija po razinama stručnosti administratora vidljivo je koliko je složenije

definirati sigurnosna pravila na distribuiranim nego na jedinstvenom vatrozidu. Tako je kod iskusnih stručnih administratora vjerojatnost anomalije porasla s 8% na 18%, kod srednje stručnih administratora s 13% na 26%, a kod početnika s 27% na 39%. Situacija je još složenija ukoliko sigurnosne politike definiraju različiti administratori, možda čak i iz različitih organizacija [15].

Na temu optimiranja distribuiranih sigurnosnih politika predstavljena su i druga rješenja pa su tako Ricardo Oliviera, Sihyung Lee i Hyong Kim [49] predstavili svoju implementaciju optimizatora – Prometheus u koji je uklopljeno i analiziranje informacija o dinamičkom usmjeravanju (engl. *routing*), što je po njima ključno za dobivanje kompletne slike o mreži.

Grupa autora predvođena Lihua Yuanom predstavila je svoju implementaciju alata za optimiranje – FIREMAN [60], koji može biti korišten za optimiranje jedinstvenog vatrozida, ali i distribuiranih sigurnosnih politika na većem broju vatrozida. Baziran je na korištenju binarnih dijagrama odluke [34] te uz standardna uklanjanja anomalija obavlja i spajanje (sažimanje) većeg broja sigurnosnih pravila u jedno, ali i osnovnu provjeru sigurnosne politike (propuštanje nesigurnih protokola i sl.).

Postoje i druge metode optimiranja distribuiranih sigurnosnih pravila [18], [62], ali budući da to nije u opsegu ovog rada, one nisu detaljno analizirane.

5. Sigurnosna pravila - Iptables vatrozid

Za potrebe ovog rada odabran je Iptables, besplatni Linux vatrozid otvorenoga programskog koda koji je dio NefFilter projekta [47]. Iptables je nasljednik ranijeg Ipchains alata raspoloživog na Linux 2.2 jezgrama dok je Iptables raspoloživ na Linux jezgrama 2.4 i 2.6. Odabran je iz razloga što korisnicima nudi najveću razinu fleksibilnosti kod konfiguriranja mrežnog filtriranja.

Iptables grupira naredbe, tj. sigurnosna pravila u tzv. lance (engl. *chains*) pri čemu svaki lanac predstavlja jednu sigurnosnu politiku. Sigurnosne politike definirane su za različite tokove mrežnog prometa:

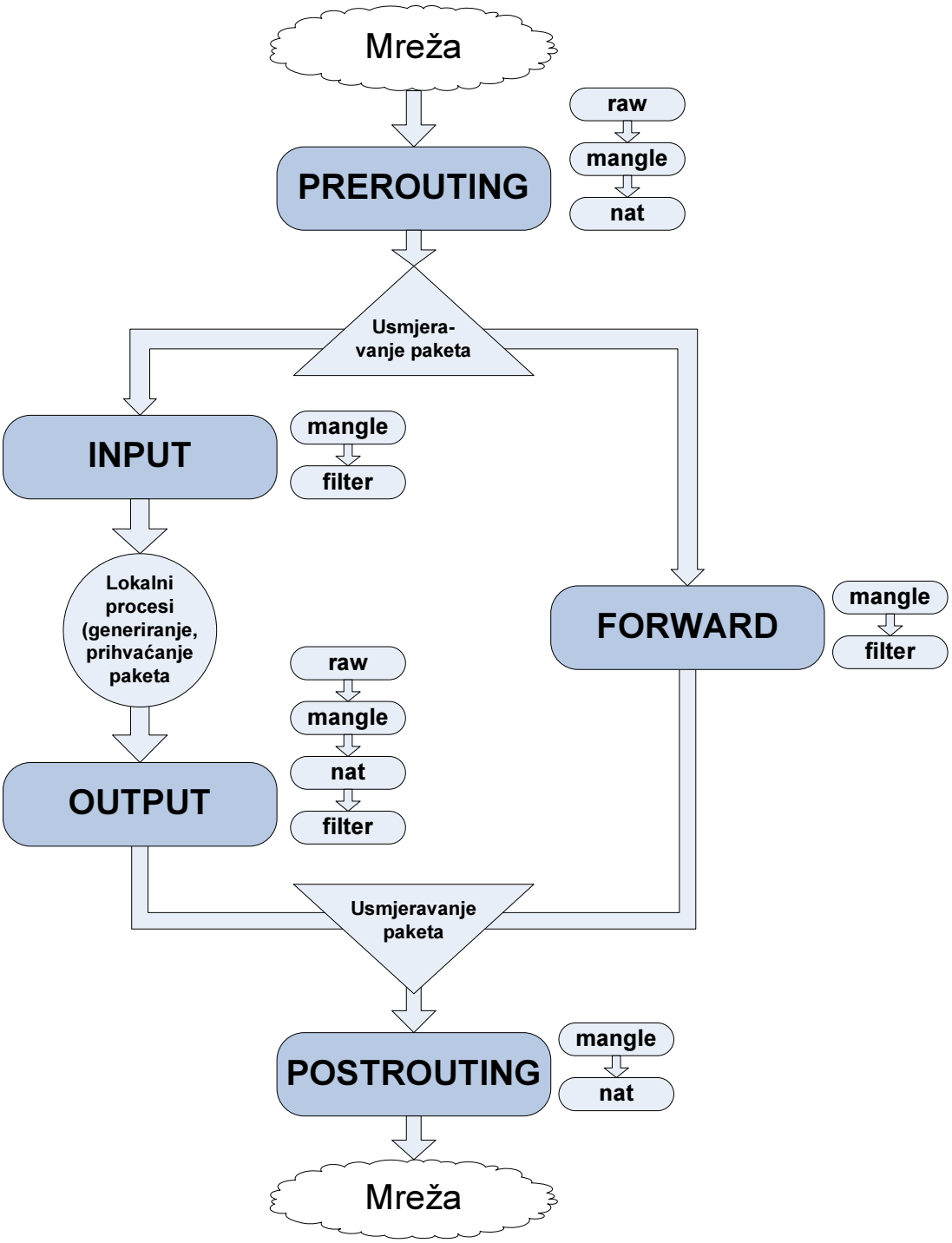
- Dolazni promet – promet koji je usmjeren izravno na vatrozid. U tu svrhu Iptables koristi **INPUT** lanac u koji se spremaju odgovarajuća sigurnosna pravila.
- Odlazni promet – promet koji generira vatrozid. Iptables naredbe za taj promet sprema u **OUTPUT** lanac.
- Prosljeđivani promet – promet koji se prosljeđuje preko vatrozida. Mrežni promet prima se preko jednog mrežnog sučelja te se može proslijediti preko tog istog sučelja ili preko nekog drugog sučelja. Iptables naredbe za tu vrstu prometa spremaju se u **FORWARD** lanac.

Osim navedene tri grupe, Iptables omogućava kreiranje i sigurnosnih politika za mrežne promete na koje se primjenjuje NAT (engl. *Network Address Translation*) mehanizam. U tu svrhu moguće je kreirati zasebne lance naredbi za DNAT (engl. *Destination NAT*), SNAT (engl. *Source NAT*) ili maskiranje (engl. *Masquerading*).

Korištenjem DNAT mehanizma ciljna se IP adresa pristiglog mrežnog prometa zamjenjuje s IP adresom računala koje se nalazi skriveno iza vatrozida te se mrežni paket preusmjerava prema tom računalu. DNAT se npr. može koristiti kod poslužitelja skrivenih iza vatrozida pri čemu ti poslužitelji koriste privatne IP adrese koje se ne smiju adresirati na Internetu; zbog toga vatrozid za te poslužitelje koristi ili jednu zasebnu javnu IP adresu ili sav promet pristigao na određeni port (ili portove) preusmjerava na njih. Za naredbe DNAT mehanizma koristi se **PREROUTING** lanac. Svi mrežni paketi koji dolaze na računalo ili se prosljeđuju moraju se usporediti s pravilima definiranim u ovom lancu.

SNAT i *Masquerading* koriste se za promjenu izvorne IP adrese mrežnih paketa. To je slučaj kad je potrebno omogućiti računalima u privatnoj mreži, koja koriste privatne IP adrese, pristup Internetu. U tom slučaju vatrozid pristiglim mrežnim paketima zamjenjuje izvornu IP adresu sa točno definiranom javnom IP adresom (SNAT) ili s IP adresom koja je vatrozidu dodijeljena na mrežnom sučelju preko kojeg je vatrozid spojen na Internet (*Masquerading*). Za naredbe SNAT i *Masquerading* mehanizma koristi se

POSTROUTING lanac. Svi mrežni paketi koje računalo generira ili koji se prosljeđuju moraju se usporediti s pravilima definiranim u ovom lancu.



Slika 12: Osnovna shema usmjeravanje mrežnih paketa pomoću Iptables alata (Iptables tutorial, http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html, 1. veljače 2011)

Na prethodnoj slici prikazana je osnovna shema obrade mrežnih paketa. Mrežni paket kroz svaki od navedenih karika lanca prolazi i kroz određene tablice (*engl*. table). Na Iptables vatrozidu postoje 4 tipa tablica:

- **Raw** – prvenstveno služi za označavanje mrežnih paketa prije nego što budu grupirani po uspostavljenim konekcijama;
- **Mangle** - koristi se za promjenu karakteristika i oznaka paketa i konekcija (npr. TOS, TTL, specijalne „*mark*“ oznake ili sigurnosne „*secmark*“ oznake);
- **Filter** – primarna svrha je prihvaćanje ili odbacivanje mrežnih paketa;
- **Nat** – za potrebe promjena ciljnih i izvornih mrežnih adresa pri čemu se samo prvi paketi u konekciji provjeravaju.

Slika 12 prikazuje kako izgleda procesuiranje mrežnih paketa preko predefiniranih i ugrađenih lanaca na Iptables vatrozidu. Ali korisnici mogu kreirati i vlastite lance te proizvoljno preusmjerivati obradu mrežnih paketa preko njih.

Iptables sigurnosna pravila sastoje se od nekoliko osnovnih elemenata:

```
iptables [–t tablica] lanac [parametri_usporedbe] [-j akcija
[parametri_akcije]]
```

Ukoliko se ne upiše polje `[-t tablica]` za INPUT, OUTPUT i FORWARD lance radi o *filter* tablici, a za PREROUTING ili POSTROUTING lance radi se o *nat* tablici. Polje `lanac` nužno je; ono može biti neko od 5 predefiniranih lanaca ili bilo koji korisnički definirani lanac. Parametri usporedbe `[parametri_usporedbe]` nisu obavezni i ukoliko niti jedan nije unesen, sigurnosno pravilo označava sve mrežne pakete. Na kraju nije nužna niti akcija `[-j akcija]`, ali onda takvo pravilo nema nikakvu svrhu, osim možda komentatorske. Na kraju, nakon akcije mogu se definirati dodatni parametri `[parametri_akcije]` dozvoljeni za tu akciju, npr. DNAT IP adrese.

Kod Firo programske implementacije za potrebe optimizacije korišten je set sigurnosnih pravila koji se dobiva kao rezultat `iptables-save` naredbe. Kod takvog izlaza naredbe se nalaze razdijeljene, prvo po tablicama (`nat`, `filter`, `mangle`, `raw`), a zatim po predefiniranim i korisnički definiranim lancima, kojima je pri tome definirana i osnovna akcija (prihvaćanja ili odbacivanja). Zog toga se iz pojedine naredbe izostavljaju elementi `iptables` i `[-t tablica]`, a naredba se označuje s oznakom –A (engl. ***A**ppend* - pridodati) na temelju čega se definirana naredba priključuje na kraj liste naredbi unutar pripadnog lanca.

```
*nat
:PREROUTING ACCEPT [700:95888]
-A PREROUTING –i eth0 –j DNAT --to-destination 192.168.1.1

*filter
:INPUT DROP [64879:14069661]
-A INPUT –p tcp –-dport 80 –j ACCEPT
```

```
-A INPUT –p tcp –-dport 8080 –j ACCEPT
```

5.1. Parametri usporedbe Iptables sigurnosnih pravila

Svaki parametar usporedbe sigurnosnog pravila identificira jednu karakteristiku mrežnog prometa. U sljedećoj tablici prikazani su neki od mogućih parametara te primjer njihovog korištenja.

Parametri koji nemaju prefiks –m (modul), dio su osnovnog Iptables programa, dok se drugi uključuju kao moduli korištenjem spomenutog prefiksa.

Tablica 12: Parametri usporedbe Iptables vatrozida

Osnovni parametri usporedbe	
Oznaka	**Opis**
`-p`	Protokol mrežnog paketa (TCP, ICMP, UDP,...). Oznaka ALL označava sve protokole. Dozvoljena inverzija, tj. negacija (`!`).
	Primjeri: `-p TCP; -p ALL; ! -p UDP`
`-s`	Izvorna IP adresa ili mreža mrežnog paketa. Dozvoljena inverzija (`!`) i *wildcard* oznake.
	Primjeri: `-s 1.1.1.1; ! -s 1.0.0.0/8; -s 2.0.0.2/255.0.0.255`
`-d`	Ciljna IP adresa ili mreža mrežnog paketa. Dozvoljena inverzija (`!`) i *wildcard* oznake.
	Primjeri: `-d 2.2.2.2; ! -d 2.0.0.0/255.0.0.0; -d 2.0.0.2/255.0.0.255`
`-i`	Sučelje preko kojeg mrežni paket dolazi na vatrozid. Iz jasnih razloga nije primjenjivo za OUTPUT lanac. Dozvoljena inverzija (`!`) i specificiranje svih sučelja istog tipa s npr. `eth+`.
	Primjeri: `-i eth0; ! -i eth+`
`-o`	Sučelje preko kojeg mrežni paket odlazi sa vatrozida. Iz jasnih razloga nije primjenjivo za INPUT lanac. Dozvoljena inverzija (`!`) i specificiranje svih sučelja istog tipa s npr. `eth+`.
	Primjeri: `-o eth0; ! -o eth+`
`-f`	Fragmentirani paketi (nakon prvog dijela). Inverzija (`!`) označava samo prve dijelove fragmentiranih paketa kao i nefragmentirane pakete.
	Primjeri: `-f; ! –f`
TCP / UDP / SCTP zajednički parametri usporedbe	

Oznaka	Opis
`--sport`	Izvorni port ili raspon portova. Dozvoljena inverzija (`!`).
	Primjeri: `--sport 1025; ! --sport 10:20`
`--dport`	Ciljni port ili raspon portova. Dozvoljena inverzija (`!`).
	Primjeri: `--dport 80; ! --dport 81:1024`
`--sports`	Izvorni niz portova. Maksimalno dozvoljeno 15 portova. Inverzija (`!`) nije dozvoljena.
	Primjeri: `-m multiport --sports 1025,1033,2045,3066`
`--dports`	Ciljni niz portova. Maksimalno dozvoljeno 15 portova. Inverzija (`!`) nije dozvoljena.
	Primjeri: `-m multiport --dports 22,21,80`
`--ports`	Izvorni i ciljni niz portova. Maksimalno dozvoljeno 15 portova. Inverzija (`!`) nije dozvoljena.
	Primjeri: `-m multiport --ports 22,21,80`
TCP parametri usporedbe	
Oznaka	**Opis**
`--tcp-flags`	TCP zastavice (dio TCP zaglavlja paketa), pri čemu se za definirane zastavice (prvo navedene) definira moraju li biti postavljene (vrijednost 1) ukoliko se nakon toga ponovno navedu ili ne (vrijednost 0) ukoliko se ne navedu ponovno. Moguće zastavice su SYN, ACK, FIN, RST, URG, PSH, ali moguće je i korištenje oznaka ALL i NONE za označavanje svih i niti jedne. Dozvoljena inverzija (`!`).

	Primjeri: `-p tcp --tcp-flags SYN,ACK,FIN SYN;`
	`-p --tcp-flags ALL SYN,ACK;`
	`-p ! –tcp-flags ALL NONE`

`--tcp-option`	TCP opcija (dio TCP zaglavlja paketa). Dozvoljena inverzija (`!`).
	Primjeri: `-p tcp --tcp-option 16;` `-p tcp ! –tcp-option 8`
ICMP parametri usporedbe	
Oznaka	**Opis**
`--icmp-type`	ICMP tip (dio ICMP zaglavlja paketa) – pri čemu se koriste brojčane oznake. Dozvoljena inverzija (`!`).
	Primjeri: `-p icmp --icmp-type 8;` `-p icmp ! --icmp-type 11`
SCTP parametri usporedbe	

Oznaka	Opis
`--chunk-types`	*Chunk* tip SCTP paketa. Nakon oznake navodi se prvo `all`, `any` ili `none`, a potom blok (*chunk*) ili niz blokova. `all`, `any` i `none` definiraju trebaju li se uspoređivati svi, bilo koji ili niti jedan blok koji slijedi. Nakon pojedinog bloka dozvoljeno je dodati oznaku ':' te definirati i posebne zastavice koje su specifične za pojedini blok (veliko slovo zastavice znači da mora biti postavljena, a malo da ne smije). Dozvoljena inverzija (`!`).
	Primjeri: `-p sctp --chunk-types any INIT,INIT_ACK;` `-p sctp ! --chunk-types any DATA:Be`
Parametri usporedbe po rasponu IP adresa	
Oznaka	**Opis**
`--src-range`	Raspon izvornih IP adresa. Dozvoljena inverzija (`!`).
	Primjeri: `-m iprange --src-range 1.1.1.1-2.2.2.2`
`--dst-range`	Raspon ciljnih IP adresa. Dozvoljena inverzija (`!`).
	Primjeri: `-m iprange ! –dst-range 3.3.3.3-4.4.4.4`
Parametri usporedbe po tipu adrese	
Oznaka	**Opis**
`--src-type`	Tip izvorne adrese. Dozvoljena inverzija (`!`). Moguće vrijednosti: ANYCAST, BLACKHOLE, BROADCAST, LOCAL, MULTICAST, NAT, PROHIBIT, THROW, UNICAST, UNREACHABLE, UNSPEC, XRESOLVE

	Primjeri:
	`-m addrtype --src-type ANYCAST,MULTICAST;`
	`-m addrtype ! --src-type NAT`

`--dst-type`	Tip ciljne adrese. Dozvoljena inverzija (`!`). Moguće vrijednosti iste kao i kod tipa izvorne adrese.
	Primjeri: `-m addrtype --dst-type ANYCAST,MULTICAST;` `-m addrtype ! --dst-type MULTICAST`
`--limit-iface-in`	Ograničenje provjere tipa adrese po dolaznom mrežnom sučelju. Moguće korištenje na PREROUTING, FORWARD i INPUT lancima.
	Primjeri: `-m addrtype –-limit-iface-in eth0`
`--limit-iface-out`	Ograničenje provjere tipa adrese po odlaznom mrežnom sučelju. Moguće korištenje na POSTROUTING, FORWARD i OUTPUT lancima.
	Primjeri: `-m addrtype –-limit-iface-out eth0`
DSCP parametri usporedbe	

Oznaka	Opis
`--dscp`	Određuje DSCP (engl. *Differentiated Services Code Point*) vrijednost u heksadecimalnom obliku. Dozvoljena inverzija (`!`).
	Primjeri: `-m dscp –-dscp 0x20; -m dscp ! –-dscp 0x16`
`--dscp-class`	Određuje DSCP klasu paketa. Dozvoljene vrijednosti BE, EF, AFxx ili CSx. Te druge definirane RFC2638 dokumentom.
	Primjeri: `-m dscp –-dscp-class BE`
Parametri usporedbe kod upravljanja zagušenjima	
Oznaka	**Opis**
`--ecn-tcp-cwr`	CWR (engl. *Congestion Window Received*) bit koji se koristi za označavanja zagušenja (ENC – *Explicit Congestion Notification*) na koje je poduzeta određena akcija. Dozvoljena je inverzija (`!`) koja zahtjeva da taj bit nije postavljen.
	Primjeri: `-m ecn --ecn-tcp-cwr;` `-m ecn ! –ecn-tcp-cwr`
`--ecn-tcp-ece`	ECE (ECN *Echo*) bit se koristi kao zahtjev za poduzimanje određene akcije (smanjenja prometa) zbog uočenog zagušenja (na zagušenje upozorava usmjerivač slanjem paketa s postavljenim EC – *Explicit Congestion* bitom). Dozvoljena je inverzija (`!`) koja zahtjeva da taj bit nije postavljen.
	Primjeri: `-m ecn --ecn-tcp-ece;` `-m ecn ! --ecn-tcp-ece`
`--ecn-ip-ect`	ECT (ECN *Capable Transport*) kod. Moguće vrijednosti su 0 (ECT=0, CE=0), 1 (ECT=0, CE=1), 2 (ECT=1, CE=0), 3 (ECT=01 CE=1). Dozvoljena je inverzija (`!`).
	Primjeri: Primjeri: `-m ecn --ecn-ip-ect 1;` `-m ecn ! --ecn-ip-ect 2`
IPSEC AH/ESP parametri usporedbe	
Oznaka	**Opis**
`--ahspi`	AH SPI (*Security Parameter Index*) broj ili raspon brojeva.
	Primjeri: `-m ah --ahspi 500; -m ah --ahspi 500:520`
`--espspi`	ESP SPI (*Security Parameter Index*) broj ili raspon brojeva.
	Primjeri: `-m esp --esppi 500;` `-m esp --espspi 200:220`
IPSEC parametri usporedbe	
Oznaka	**Opis**

`--dir`	Identificira *policy* za „enkapsulaciju“ ili „dekapsulaciju“. Vrijednosti: *in* i *out*.
	Primjeri: `-m policy --dir in; -m policy --dir out`
`--pol`	Definira je li paket namijenjen za IPSec obradu. Vrijednosti: *ipsec* i *none*.
	Primjeri: `-m policy --pol ipsec;` `-m policy --pol none`
`--strict`	Striktno uspoređivanje po definiranom *policy*-u.
	Primjeri: `-m policy –-strict`
`--reqid`	*ReqID policy*-a. Dozvoljena inverzija (`!`).
	Primjeri: `-m policy --reqid 373;` `-m policy ! --reqid 4778`
`--spi`	SPI (engl. *Security Parameters Index*) sigurnosne organizacije. Dozvoljena inverzija (`!`).
	Primjeri: `-m policy --spi 2147483648;` `-m policy ! --spi 1073741824`
`--proto`	Enkapsulacijski protokol (*ah*, *esp*, *ipcomp*). Dozvoljena inverzija (`!`).
	Primjeri: `-m policy --proto ah;` `-m policy ! --proto esp`
`--mode`	Enkapsulacijski mod (*tunnel*, *transport*). Dozvoljena inverzija (`!`).
	Primjeri: `-m policy --mode tunnel;` `-m policy ! –-mode transport`
`--tunnel-src`	Kad se koristi tunelirani mod, krajnja izvorna IP adresa (ili mreža) tunela. Dozvoljena inverzija (`!`).
	Primjeri: `-m policy –-mode tunnel –-tunnel-src 1.1.1.1;` `-m policy –-mode tunnel ! –-tunnel-src 1.1.1.0/24`
`--tunnel-dst`	Kad se koristi tunelirani mod, krajnja ciljna IP adresa (ili mreža) tunela. Dozvoljena inverzija (`!`).
	Primjeri: `-m policy –-mode tunnel –-tunnel-dst 1.1.1.1;` `-m policy –-mode tunnel ! –-tunnel-dst 1.1.1.0/24`
`--next`	Definira novi element u specifikaciji *policy*-a. Dozvoljen uz `--strict`.
	Primjeri: `-m policy --strict --next;`

Parametri usporedbe definiranih oznaka (*mark*)	
Oznaka	**Opis**
`--mark`	Oznaka postavljena u sklopu MARK ili CONNMARK akcije. Moguće vrijednosti su od 0-4294967296, ali može se postaviti i maska (npr. 1/1 označava sve neparne oznake). Dozvoljena inverzija (`!`).
	Primjeri: `-m mark --mark 0x15;` `-m connmark ! --mark 0x15/0x15`
Parametri usporedbe stanja konekcija	
Oznaka	**Opis**
`--state`	Status konekcije. Može biti u jednoj od 5 vrijednosti: INVALID, ESTABLISHED, RELATED, UNTRACKED i NEW. Inverzija je dozvoljena (`!`), ali se prilikom spremanja u listu naredba sprema bez oznake inverza – s preostalim vrijednostima.
	Primjeri: `-m state --state RELATED,ESTABLISHED`
Vremenski parametri usporedbe	
Oznaka	**Opis**
`--datestart`	Pravilo usporedbe vrijedi samo nakon definiranog datuma. Format: *YYYY*[**-***MM*[**-***DD*[**T***hh*[**:***mm*[**:***ss*]]]]]
	Primjer: `-m time --datestart 2010-02-01T14:00:30`
`--datestop`	Pravilo usporedbe vrijedi samo do definiranog datuma. Format: *YYYY*[**-***MM*[**-***DD*[**T***hh*[**:***mm*[**:***ss*]]]]]
	Primjer: `-m time --datestop 2012-01-01T14:30:30`
`--timestart`	Pravilo usporedbe vrijedi dnevno samo nakon definiranog vremena. Format: *hh:mm[:ss]*
	Primjeri: `-m time --timestart 08:00`
`--timestop`	Pravilo usporedbe vrijedi dnevno samo do definiranog vremena. Format: *hh:mm[:ss]*
	Primjeri: `-m time --timestop 16:00`
`--utc`	Definirane prethodne 4 vrijednosti smatraju se UTC
	Primjeri: `-m time --utc`
`--localtz`	Definirane prve 4 vrijednosti smatraju se LTZ
	Primjeri: `-m time --localtz`

`--monthdays`	Pravilo vrijedi samo za definirane dane u mjesecu (1-31). Dozvoljena inverzija (`!`).
	Primjeri: `-m time –-monthdays 1,2,3;` `-m time ! –-monthdays 21,22`
`--weekdays`	Pravilo vrijedi samo za definirane dane u tjednu (Mo, Tu, We, Th, Fr, Sa, Su). Dozvoljena inverzija (`!`).
	Primjeri: `-m time –-weekdays Mo,Tu;` `-m time ! –-weekdays Sa,Su`
Statistički parametri usporedbe	
Oznaka	**Opis**
`--mode`	Statistički mod. Može biti *random* ili *nth*.
	Primjeri: `-m statistics –-mode random;` `-m statistics –-mode nth`
`--probability`	Vjerojatnost uspoređivanja paketa slučajnim odabirom Vrijednosti od 0 do 1. Dozvoljen uz *random* mod.
	Primjeri: `-m statistics –-mode random –-probability 0.5`
`--every`	Uspoređuje svaki n-ti paket. Dozvoljen uz *nth* mod.
	Primjeri: `-m statistics –-mode nth –-every 5`
`--packet`	Postavlja inicijalni brojač za *nth* mod.
	Primjeri: `-m statistics –-mode nth –-packet 1`
Parametri usporedbe konekcije (izravniji pristup)	
Oznaka	**Opis**
`--ctstate`	Status konekcije (detaljniji i izravniji pristup od prethodnog) preko CONNTRACK-a. Može biti u jednoj od 7 vrijednosti: INVALID, ESTABLISHED, RELATED, UNTRACKED, NEW, DNAT i SNAT. Dozvoljena inverzija (`!`).
	Primjeri: `-m conntrack ! –-ctstate RELATED`
`--ctproto`	Kao i `–p`. Protokol mrežnog paketa (TCP, ICMP, UDP,...). Oznaka ALL označava sve protokole. Dozvoljena inverzija (`!`).
	Primjeri: `-m conntrack ! ––ctproto TCP`
`--ctorigsrc`	Originalna izvorna IP adresa ili mreža od *conntrack* zapisa na koji se paket odnosi. Dozvoljena inverzija (`!`) i *wildcard* oznake.
	Primjeri: `-m conntrack ––ctorigsrc 1.1.1.1;` `-m conntrack ! ––ctorigsrc 1.0.0.0/8`

`--ctorigdst`	Originalna ciljna IP adresa ili mreža od *conntrack* zapisa na koji se paket odnosi. Dozvoljena inverzija (`!`) i *wildcard* oznake.
	Primjeri: `-m conntrack –ctorigsdst 2.2.2.2;` `-m conntrack ! ––ctorigdst 2.0.0.2/255.0.0.255`
`--ctreplsrc`	Originalna izvorna IP adresa ili mreža dobivena iz *conntrack* odgovora. Dozvoljena inverzija (`!`) i *wildcard* oznake.
	Primjeri: `-m conntrack ––ctreplsrc ! 1.1.1.1;` `-m conntrack ! ––ctreplsrc 1.0.0.0/8`
`--ctrepldst`	Originalna ciljna IP adresa ili mreža dobivena iz *conntrack* odgovora. Dozvoljena inverzija (`!`).
	Primjeri: `-m conntrack ––ctrepldst 1.1.1.1;` `-m conntrack ! ––ctrepldst 1.0.0.0/255.0.0.255`
`--ctorigsrcport`	Originalni izvorni port od *conntrack* zapisa na koji se paket odnosi. Dozvoljena inverzija (`!`).
	Primjeri: `-m conntrack ––ctorigsrcport 1025;` `-m conntrack ! ––ctorigsrcport 22`
`--ctorigdstport`	Originalni ciljni port od *conntrack* zapisa na koji se paket odnosi. Dozvoljena inverzija (`!`).
	Primjeri: `-m conntrack ! ––ctorigsdstport 12000;` `-m conntrack ––ctorigdstport 21`
`--ctreplsrcport`	Originalni izvorni port dobiven iz *conntrack* odgovora. Dozvoljena inverzija (`!`).
	Primjeri: `-m conntrack ! ––ctreplsrcport 1025;` `-m conntrack ––ctreplsrcport 8080`
`--ctrepldstport`	Originalni ciljni port dobiven iz *conntrack* odgovora. Dozvoljena inverzija (`!`).
	Primjeri: `-m conntrack ! ––ctrepldstport 1025;` `-m conntrack ––ctrepldstport 80`
`--ctstatus`	Status konekcije *conntrack*. Moguće vrijednosti: NONE (konekcija nema status), EXPECTED (konekcija očekivana), SEEN_REPLY (dobiven odgovor, ali još nije potvrđena), ASSURED (potvrđena konekcija, neće se uklanjati dok ne istekne ili ju jedna strana eksplicitno ne zatvori), CONFIRMED (konekcija je potvrđena). Dozvoljena inverzija (`!`).
	Primjeri: `-m conntrack ––ctstatus NONE,EXPECTED;` `-m conntrack ! ––ctstatus ASSURED`
`--ctexpire`	Broj sekundi dok conntrack konekcija ne istekne. Dozvoljen raspon i inverzija (`!`). Maksimalna vrijednost 4294967295.
	Primjeri: `-m conntrack ––ctexpire 100:200;` `-m conntrack ! ––ctexpire 100:200`

Parametri usporedbe veličine paketa i konekcije	
Oznaka	**Opis**
`--connbytes`	Označava ukupni ili prosječni, broj ili raspon paketa ili okteta (engl. *bytes*) koji su do procesuiranog paketa zabilježeni u sklopu pripadajuće konekcije. Dozvoljena inverzija (`!`).
	Primjeri: `-m connbytes --connbytes 1000:10000;` `-m connbytes ! --connbytes 100:200`
`--connbytes-dir`	Označava koji se paketi promatraju. Moguće vrijednosti: *original*, *reply*, *both*.
	Primjeri: `-m connbytes –-connbytes-dir original`
`--connbytes-mode`	Definira što se mjeri, broj paketa (*packets*), broj okteta (*bytes*), ili prosječan broj paketa (*avgpkt*).
	Primjeri: `-m connbytes –-connbytes-mode bytes`
Parametri usporedbe po vlasničkim podacima generiranih paketa (primjenjivo samo na OUTPUT i POSTROUTING lancima)	
Oznaka	**Opis**
`--uid-owner`	Vlasnik generiranog paketa (korisnik koji je pokrenuo proces koji je generirano paket). Dozvoljena inverzija (`!`).
	Primjeri: `-m owner –-uid-owner 0;` `-m owner ! –-uid-owner 504`
`--gid-owner`	Grupa vlasnika generiranog paketa (korisnik koji je pokrenuo proces koji je generirano paket). Dozvoljena inverzija (`!`).
	Primjeri: `-m owner –-gid-owner 500;` `-m owner ! –-gid-owner 0`
`--socket-exists`	Provjera je li paket povezan sa priključnicom (engl. *socket*). Dozvoljena inverzija (`!`).
	Primjeri: `-m owner --socket-exists;` `-m owner ! --socket-exists`
***Limit* parametri usporedbe**	
Oznaka	**Opis**
`--limit`	Ograničavanje broja paketa koji se uspoređuju po jedinici vremena (sekunda, minuta, sat, dan) – maksimalna prosječna vrijednost
	Primjeri: `-m limit –-limit 10/second`
`--limit-burst`	Ograničavanje broja paketa koji se uspoređuju po jedinici vremena (sekunda, minuta, sat, dan) – maksimalna inicijalna vrijednost, kad se dostigne ne dozvoljavaju se novi paketi
	Primjeri: `-m limit –-limit-burst 5`

Parametri usporedbe broja konekcija	
Oznaka	**Opis**
`--connlimit -above`	Provjerava je li broj konekcija veći od definiranog broja. Dozvoljena inverzija (`!`).
	Primjeri: `-m connlimit --connlimit-above 2;` `-m connlimit ! --connlimit-above 2`
`--connlimit-mask`	Mrežna maska (0-32 za ipv4, 0-128 za ipv6) kojom se broj konekcija limitira na mrežu.
	Primjeri: `-m connlimit --connlimit-mask 24;`
***Hashlimit* parametri usporedbe (naprednija verzija ograničavanja broja paketa koja za svako pravilo generira *hash* tablicu)**	
Oznaka	**Opis**
`--hashlimit-upto`	Ograničavanje broja paketa koji se uspoređuju po jedinici vremena (sekunda, minuta, sat, dan) – maksimalno dozvoljena vrijednost.
	Primjeri: `-m hashlimit –-hashlimit-upto 10/second`
`--hashlimit -above`	Ograničavanje broja paketa koji se uspoređuju po jedinici vremena (sekunda, minuta, sat, dan) – minimalno dozvoljena vrijednost.
	Primjeri: `-m hashlimit –-hashlimit-above 5/minute`
`--hashlimit-mode`	Definira koje vrijednosti se mogu koristiti kao *hash* vrijednosti. Mogući su: *dstip* (ciljna IP adresa), *srcip* (izvorna IP adresa), *dstport* (ciljni port), *srcport* (izvorni port).
	Primjeri: `-m hashlimit –-hashlimit-mode srcip;` `-m hashlimit –-hashlimit-mode srcip,dstport`
`--hashlimit–src -mask`	Maska koja se primjenjuje na *srcip* mod. Vrijednosti od 0 (sve IP adrese) do 32 (1 IP adresa)
	Primjeri: `-m hashlimit –-hashlimit-src-mask 24`
`--hashlimit–dst -mask`	Maska koja se primjenjuje na *dstip* mod. Vrijednosti od 0 (sve IP adrese) do 32 (1 IP adresa)
	Primjeri: `-m hashlimit –-hashlimit-dst-mask 26`
`--hashlimit-name`	Definira naziv *hash* tablice – koji će biti kasnije vidljiv kao datoteka s tim nazivom.
	Primjeri: `-m hashlimit –-hashlimit-name tmp123`
`--hashlimit -burst`	Ograničavanje broja paketa koji se uspoređuju po jedinici vremena (sekunda, minuta, sat, dan) – maksimalna inicijalna vrijednost, kad se dostigne ne dozvoljavaju se novi paketi
	Primjeri: `-m hashlimit –-hashlimit-burst 100`

`--hashlimit -htable-size`	Definira maksimalni broj limitatora za pravilo. Npr. ukoliko je moguće imati sve portove, a definirano je korištenje *dstport* kao hash vrijednosti, sljedeći uvjet će omogućiti otvaranje konekcija prema samo 50 portova.
	Primjeri: `-m hashlimit –-hashlimit-htable-size 50`
`--hashlimit -htable-max`	Definira maksimalni broj zapisa u *hash* tablici.
	Primjeri: `-m hashlimit –-hashlimit-htable-max 500`
`--hashlimit -htable -gcinterval`	Definira koliko često (u milisekundama) će se *garbage collector* pokretati i oslobađati nekorištene tokene.
	Primjeri: `-m hashlimit --hashlimit-htable-gcinterval 1000`
`--hashlimit -htable -expire`	Definira tzv. *idle* vrijeme – koliko dugo je potrebno da zapis u *hash* tablici istekne.
	Primjeri: `-m hashlimit --hashlimit-htable-expire 10000`

Parametri usporedbe bazirani na prošlim (nepovezanim) paketima

Oznaka	Opis
`--name`	Kreiranje nove liste u koju se spremaju podaci o konekcijama. Predefinirano se spremaju u DEFAULT listu (što nije preporučljivo ako je zbog obujma korisno dijeljenje u više lista).
	Primjeri: `-m recent –-name tmp_list`
`--set`	Dodavanje novog zapisa (*timestamp* vremenski zapis, izvorna IP adresa iz paketa) u predefiniranu ili eksplicitno definiranu listu.
	Primjeri: `-m recent –-set`
`--rcheck`	Provjera nalazi li se izvorna adresa paketa u predefiniranoj ili eksplicitno definiranoj listi. Dozvoljena inverzija (`!`).
	Primjeri: `-m recent –-rcheck; -m recent ! –-rcheck`
`--update`	Provjera nalazi li se izvorna adresa paketa u predefiniranoj ili eksplicitno definiranoj listi pri čemu mijenja i zapisani *timestamp* vremenski zapis. Dozvoljena inverzija (`!`).
	Primjeri: `-m recent –-update; -m recent ! –-update`
`--remove`	Provjera nalazi li se izvorna adresa paketa u predefiniranoj ili eksplicitno definiranoj listi pri čemu uklanja i pronađeni zapis. Dozvoljena inverzija (`!`).
	Primjeri: `-m recent –-remove; -m recent ! –-remove`

`--seconds`	Parametar koji se koristi ili uz `--rcheck` ili uz `--update`. Definira koliko najviše sekundi smije paket stajati u listi. Dozvoljena inverzija (`!`).
	Primjeri: `-m recent --seconds 100;` `-m recent ! --seconds 500`
`--hitcount`	Parametar koji se koristi ili uz `--rcheck` ili uz `--update`. Definira koliko najmanje puta mora biti prethodno pronađen definirani tip paketa. Dozvoljena inverzija (`!`).
	Primjeri: `-m recent --rcheck --hitcount 10;` `-m recent –update ! --hitcount 20`
`--rttl`	Parametar koji provjerava je li TTL vrijednost u obrađivanom paketu jednaka kao i TTL vrijednost u originalnom (prvom zabilježenom) paketu. Dozvoljena inverzija (`!`).
	Primjeri: `-m recent --rttl; -m recent ! --rttl`
`--rsource`	Definira spremanje izvorne IP adrese i porta iz paketa u korištenu *recent* listu.
	Primjeri: `-m recent --rsource`
`--rdest`	Definira spremanje ciljne IP adrese i porta iz paketa u korištenu *recent* listu.
	Primjeri: `-m recent --rdest`
Parametri usporedbe po izvornoj MAC adresi	
Oznaka	**Opis**
`--mac-source`	MAC adresa izvorišta. Dozvoljena inverzija (`!`).
	Primjeri: `-m mac --mac-source 12:34:56:78:90:AA;` `-m mac ! --mac-source 12:34:56:78:90:AA`
Parametri usporedbe po duljini paketa	
Oznaka	**Opis**
`--length`	Duljina mrežnog paketa određena brojem ili najčešće rasponom. Dozvoljena inverzija (`!`).
	Primjeri: `-m length --length 1000;` `-m length ! --length 500:600`
Parametri usporedbe po TTL vrijednosti	
Oznaka	**Opis**
`--ttl-eq`	Točno određena TTL vrijednost u paketu. Dozvoljena inverzija (`!`).

	Primjeri: `-m ttl –-ttl-eq 50; -m ttl ! –-ttl-eq 70`
`--ttl-gt`	Niz TTL vrijednosti većih od specificirane vrijednosti.
	Primjeri: `-m ttl –-ttl-gt 60`
`--ttl-lt`	Niz TTL vrijednosti manje od specificirane vrijednosti.
	Primjeri: `-m ttl –-ttl-lt 40`
Parametri usporedbe po tipu usluge	
Oznaka	**Opis**
`--tos`	Označava TOS (engl. *Type of Service*) polje paketa i njegovu vrijednost koja može biti određena u TOS akciji. Moguće vrijednosti: *Minimize-Delay* (0x10), *Maximize-Throughput* (0x08), *Maximize-Reliability* (0x04), *Minimize-Cost* (0x02) *i Normal-Service* (0x00)
	Primjeri: `-m tos --tos 0x10`
Ostali parametri usporedbe	
Oznaka	**Opis**
`--helper`	Za identificiranje paketa koji se nalaze u vezi s postojećom konekcijom. Definirana nazivom protokola i porta na kojem se nalazi. Primjer korištenja: FTP i IRC.
	Primjeri: `-m helper –-helper ftp-21`
`--pkt-type`	Određuje tip paketa. Moguće vrijednosti su unicast, multicast, broadcast. Dozvoljena inverzija (`!`).
	Primjeri: `-m pkttype --pkt-type unicast;` `-m pkttype ! --pkt-type broadcast`
`--realm`	Određuje tip *realm* usmjeravanja. Realm je dio Linux usmjeravanja (npr. kada se koristi BGP – *Border Gateway Protocol*). Dozvoljena maska i inverzija (`!`).
	Primjeri: `-m realm -–realm 0x4;` `-m realm ! –-realm 0x4/0x4;`
`--mss`	Označava TCP maksimalnu veličinu segmenta (MSS – *Maximum Segment Size*). Može biti broj ili raspon. Dozvoljena inverzija (`!`).
	Primjeri: `-m tcpmss --mss 2000:2500;` `-m tcpmss --mss 1000; -m tcpmss ! --mss 0-2000`

Izvor: Iptables MAN (manual) stranica, 2011.

Važno je napomenuti kako se prethodno navedeni parametri uglavnom ne mogu svi međusobno kombinirati budući da neki prolaze samo s određenim lancima ili zahtijevaju postojanje točno određenih vrijednosti parametara. Također, iako su u Iptables dokumentaciji navedeni parametri definirani kao „parametri uspoređivanja“ (engl. *matching*) prilikom konfiguriranja, nisu svi tako tretirani. Naime, neki od njih obavljaju i određene akcije (primjer limitirajući parametri ili *recent* parametri koji modificiraju određene zapise ii brojače). Stoga se ti parametri promatraju kao dio akcije.

U svrhu optimiranja sigurnosnih pravila, moguće je definiranje i sigurnosnih pravila koji imaju i druge parametre, a u tom slučaju se oni bez dodatnog konfiguriranja Firo programa ne promatraju kao brojevi, IP adrese i sl., već kao obični tekstualni nizovi. I ako su u promatrana dva sigurnosna pravila takvi nizovi identični, dozvolit će se standardne operacije optimiranja (uklanjanje nepotrebnog pravila, spajanje u jedno pravilo, uklanjanje nepotrebnih parametara i elemenata parametara).

5.2. Akcije Iptables sigurnosnih pravila

Iptables sigurnosna pravila uz parametre prepoznavanja mrežnog prometa imaju u pravilu i određenu akciju. Standardne akcije su prihvaćanje (ACCEPT) i odbacivanje (DROP) mrežnog prometa. Moguće je i korištenje akcije „povratak“ (RETURN) koja služi za povratak iz definiranog korisničkog lanca ili primjenu osnovne sigurnosne politike ukoliko se pronađe unutar nekog od osnovnih Iptables lanaca. Standardna akcija je i QUEUE koja mrežni paket preusmjerava na korisnički definirane procese.

Uz osnove 4 akcije, Iptables podržava i druge akcije koje imaju različite svrhe:

- CLASSIFY - omogućava definiranje *skb->priority* (engl. skb - *socket buffer*) vrijednosti kako bi se omogućilo upravljanje prometom (CBQ – *Class Based Queueing*),
- CLUSTERIP – omogućava korištenje više računala s istom IP i MAC adresom bez *Load Balancer* uređaja ispred njih,
- MARK – postavlja Netfilter oznake (*mark*) povezane s paketom,
- CONNMARK – postavlja Netfilter oznake (*marks*) povezane s konekcijom,
- SECMARK – postavlja sigurnosne oznake iz paketa za potrebe drugih sustava (npr. SELinux),
- CONNSECMARK – postavlja sigurnosne oznake iz paketa u konekciju,
- NOTRACK – gasi praćenje konekcija za prepoznati mrežni promet,
- ECN – omogućava zaobilaženje određenih ECN rupa,
- LOG – logira informacije o mrežnom paketu,
- NFLOG – kao i LOG akcija, služi za slanje informacije o mrežnom paketu pozadinskom procesu,

- ULOG – omogućava korisnički način logiranja prepoznatih paketa,
- TRACE – označava pakete kako bi ih jezgra operacijskog sustava mogla logirati,
- RATTEST – prikuplja informacije za potrebe statistike,
- DNAT – definira promjenu ciljne IP adrese,
- SNAT - definira promjenu izvorne IP adrese,
- SAME – slično kao i SNAT/DNAT, ali klijentu za svaku konekciju daje uvijek istu izvornu ili ciljnu adresu,
- MASQUERADE – mijenja izvornu IP adresu mrežnog paketa s onom koja je dodijeljena mrežnom sučelju preko kojeg paket napušta računalo,
- MIRROR – eksperimentalna akcija, mijenja izvornu i ciljnu IP adresu,
- NETMAP – statički mijenja IP adrese iz jedne mreže s IP adresama iz druge mreže,
- REDIRECT – preusmjerava paket na samo računalo pri čemu mijenja ciljnu IP adresu u onu koja je dodijeljena ulaznom mrežnom sučelju ili u lokalnu IP adresu (127.0.0.1),
- SET – dodaje ili uklanja zapise iz Linux IP seta,
- TCPOPTSTRIP – skida TCP opcije s TCP paketa (MSS, *timestamp*, Window Scale, MD5 potpis,...),
- DSCP – omogućava promjenu DSCP bitova unutar TOS zaglavlja paketa,
- TCPMSS – omogućava promjenu MSS (engl. *Maximum Segment Size*) vrijednosti TCP paketa sa SYN zastavicom,
- TOS – postavlja ToS broj u IPv4 zaglavlju,
- TTL – postavlja TTL broj u IPv4 zaglavlju,
- REJECT[2] - odbacuje mrežni paket uz slanje popratne *error* poruke pošiljaocu.

Iza pojedinih akcija moguće je navođenje različitih definiranih parametara koji omogućuju izvođenje neke akcije, kao npr. `-j TTL --ttl-set 50` i sl. Ali budući da to nije važno za optimizacijske procedure, oni nisu niti navedeni u ovom radu. S aspekta optimiranja sigurnosnih pravila od značaja je jedino prekidaju li te akcije obradu mrežnog paketa ili ne. Ukoliko prekidaju, tada postoji mogućnost da se iza njih nalaze druga pravila koja se mogu ukloniti. Ukoliko ne prekidaju, onda najčešće postoji samo mogućnost spajanja, a u tom slučaju nužna je istovjetnost „akcijskih“ parametara jer se spajanja obavljaju samo po parametrima „usporedbe“.

[2] Naredbe odbacivanja DROP i REJECT razlikuju se po tome što prva samo ignorira mrežni paket, dok druga šalje *error* poruku o odbacivanju mrežnog paketa izvornom računalu koje je poslalo odbačeni paket.

6. Firo – optimizator sigurnosnih pravila

Firo (***FIR****ewall* ***O****ptimizer*) je optimizator sigurnosnih pravila koji je prilagođen Iptables sigurnosnim pravilima, ali se vještim preinakama može prilagoditi i za druge vatrozide zasnovane na listama sigurnosnih pravila. Razvijen je na Fakultetu elektrotehnike i računarstva (FER), Zavod za elektroničke signale i obradu informacija (ZESOI), Laboratorij za signale i sustave (LSS), a kao programski jezik korišten je C++.

Način izvođenja Firo optimizatora je preko naredbenog retka na Linux ili Windows operacijskim sustavima. Za njega nije razvijeno grafičko sučelje jer je prvenstveno namijenjen Iptables vatrozidima koji se isto konfiguriraju preko naredbenog retka.

Tijekom svog rada Firo iterira kroz nekoliko optimizacijskih procedura u kojima generira datoteke s optimiranim listama sigurnosnih politika, tj. nizove sigurnosnih pravila pri čemu svaka verzija predstavlja jedan korak optimizacije. Zahvaljujući tim međukoracima i ukupnoj datoteci s opisom svih optimizacijskih akcija, administratori mogu jednostavno pratiti zašto su određena pravila izmijenjena ili uklonjena.

6.1. Funkcionalnosti Firo programa

Firo otkriva i uklanja anomalije pri čemu se sigurnosna politika vatrozida ne mijenja. Sve anomalije čije uklanjanje može imati efekta na ukupnu sigurnosnu politiku Firo ne detektira niti uklanja. To su anomalije korelacije i generalizacije. Osim tih dviju anomalija, Firo ne može uklanjati niti anomalije irelevantnosti zbog toga što je zasnovan samo na analizi niza sigurnosnih pravila te ne uzima u obzir nikakve druge izvore informacija.

Osim što uklanja anomalije, Firo u svrhu optimiranja obavlja i druge radnje kao što je spajanje dvaju sigurnosnih pravila u jedno jedinstveno pravilo. Kako bi se različite sigurnosne naredbe objedinile potrebno je da su im svi parametri jednaki osim jednoga (izuzetak su datum i vrijeme gdje se spajanje obavlja preko dva parametra). Spajanje različitih sigurnosnih pravila obavlja se preko većine prethodno opisanih parametara.

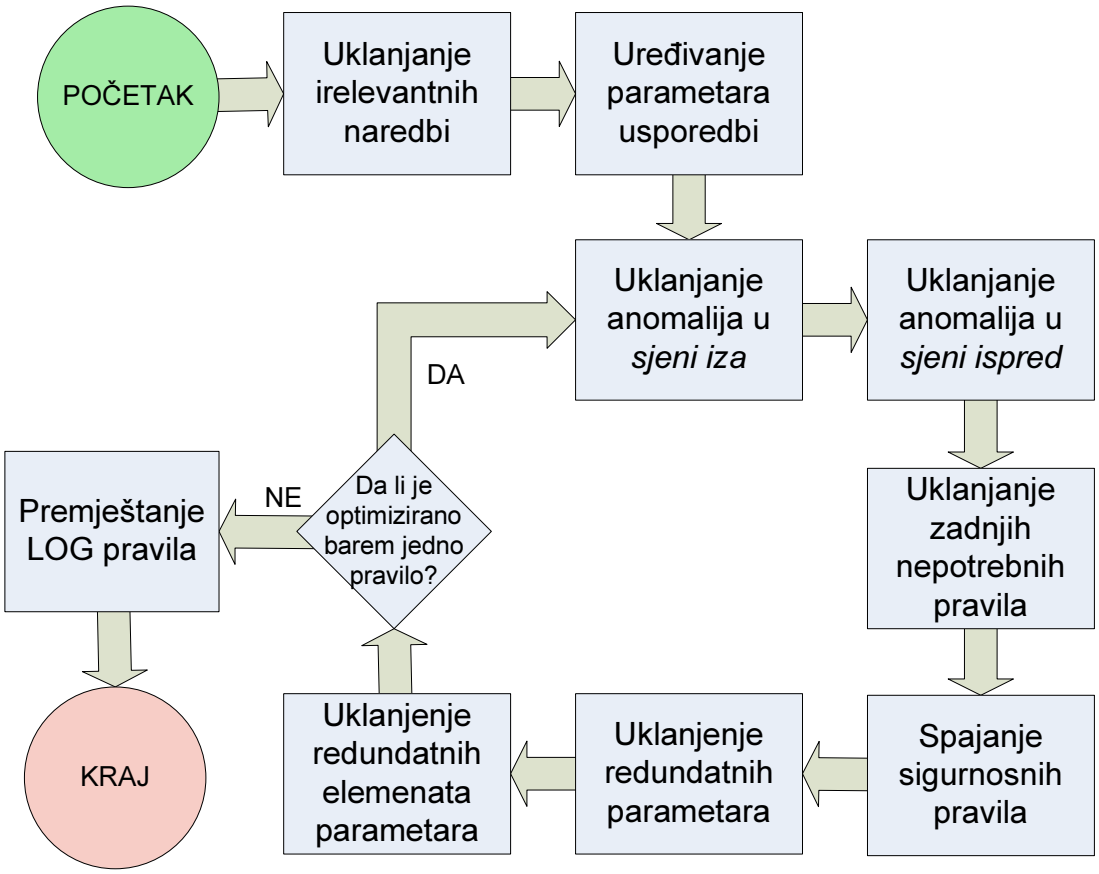
Budući da se objedinjavanjem sigurnosnih pravila mijenja njihov poredak, moguće je uslijed tih promjena prouzročiti generiranje novih anomalija pa se stoga nakon svakog procesa spajanja pravila ponovno pokreće procedura za detekciju i uklanjanje anomalija.

Za razliku od ostalih rješenja optimiranja, Firo omogućava optimizacijsku proceduru koja nije zasnovana samo na protokolu, ciljnoj i izvornoj IP adresi te ciljnom i izvornom portu. Firo uz navedene parametre uključuje i mrežna sučelja, TTL, ToS, duljinu paketa, MAC izvornu adresu, SYN zastavice,

ICMP tip, oznake za ograničavanje broja mrežnih paketa u jedinici vremena (engl. *limit*) i sve ostale parametre popisane u poglavlju 5.1. Pa čak i druge nepopisane koje prepoznaje kao znakovni niz.

Osim toga, postojeće optimizacijske metode omogućavaju optimiranje sigurnosnih pravila kojima je akcija samo „odbaci“ ili „prihvati“. Firo se konfigurira tako da optimizira pravila koja mogu imati prekidajuće ili neprekidajuće akcije. Time su obuhvaćene dvije spomenute prekidajuće akcije, ali i brojne druge kao što su npr. akcije za preusmjerivanje mrežnog prometa.

Firo pruža i posebne optimizacijske procedure za sigurnosna pravila koja obavljaju logiranje mrežnog prometa. Takva pravila u osnovi nisu poželjna za performanse jer najčešće zahtijevaju zapisivanje na disk ili u boljem slučaju slanje log zapisa na udaljeno računalo na kojem se obavlja pohranjivanje.



Slika 13: Proces optimiranja Firo programa

6.1.1. Uklanjanje irelevantnih naredbi

Na samom početku program jednokratno uklanja jednostavno uočljiva irelevantna pravila. Iz samih pravila je vidljivo kako se ona nikada ne realiziraju. Takvih „pogrešaka“ može biti neograničeno mnogo pa je zato

optimiranje usmjereno samo na one pogreške koje Iptables alat (v 1.4.10) dozvoljava.

Primjerice to može biti pravilo bez ikakve definirane akcije (nedostaje `–j <action>`):

```
–A INPUT –p TCP
```

Dodatno, to mogu biti i pravila kojima parametri nisu dobro definirani:

- pravilo s invertiranim maksimalnim rasponom portova

```
–A INPUT –p TCP ! --sport 0:65535 –j ACCEPT
```

- pravilo s krivim rasponom IP adresa

```
–A INPUT –p TCP -m iprange --src-range 2.2.2.2-
1.1.1.1 –j ACCEPT
```

- pravilo s invertiranim maksimalnim rasponom IP adresa

```
–A INPUT –p TCP ! –d 0.0.0.0/0 –j ACCEPT
```

- pravilo s krivo definiranim TTL vrijednostima

```
–A INPUT -m ttl –-ttl-gt 255 –j ACCEPT;
–A INPUT -m ttl –-ttl-lt 0 –j ACCEPT
```

- pravilo s invertiranim svim raspoloživim znakovnim vrijednostima

```
-A INPUT -m addrtype ! --src-type
UNSPEC,UNICAST,LOCAL,BROADCAST,ANYCAST,MULTICAST
,BLACKHOLE,UNREACHABLE,PROHIBIT,THROW,NAT,XRESOL
VE –j ACCEPT
```

6.1.2. Uređivanje parametara usporedbi

Nakon uklanjanja irelevantnih naredbi slijedi jednokratno uređivanje parametara u sklopu ispravnih naredbi. Uređivanje je usmjereno na 3 tipa podataka - parametara:

- Port i broj – ukoliko su definirani kao niz (engl. *array*), a dozvoljen je zapis u obliku raspona, tada se nastoje konvertirati u raspon. Ukoliko je niz brojeva slučajno dan u ne-sortiranom poretku tada se oni sortiraju

```
–A INPUT –p TCP –m multiport --sports 2,3,4,5,6
–j ACCEPT
```

se zamjenjuje s

```
–A INPUT –p TCP --sport 2:6 –j ACCEPT
```

- IP adrese – ukoliko su definirane kao raspon (engl. *range*), a moguć je zapis u obliku mreže s mrežnom maskom, nastoji se obaviti konverzija.

```
–A INPUT –p TCP –m iprange ! --src-range
1.2.3.0-1.2.3.255 –j ACCEPT
```

se zamjenjuje s

```
–A INPUT –p TCP ! –s 1.2.3.0/24 –j ACCEPT
```

- Znakovni niz – ukoliko se sastoji od niza vrijednosti, kojih postoji samo ograničeni skup te je dozvoljena inverzija, niz se konvertira iz inverznog u normalni ili obrnuto ukoliko je tako dobiven niz kraći.

```
-A INPUT -m addrtype ! --src-type
UNSPEC,UNICAST,LOCAL,BROADCAST,ANYCAST,MULTICAST
,BLACKHOLE,UNREACHABLE,PROHIBIT,THROW –j ACCEPT
```

se zamjenjuje s

```
-A INPUT -m addrtype --src-type NAT,XRESOLVE
–j ACCEPT
```

6.1.3. Uklanjanje anomalije u „sjeni iza“

Anomalija u „sjeni iza“ (engl. *Shadowed after*) već je definirana u poglavlju 3.1 kao kombinacija sigurnosnih pravila gdje prvo pravilo (u definiranom poretku) ima prekidajuću akciju i označava sav mrežni promet koji označava i drugo pravilo, tj. mrežni promet koji označava drugo pravilo podskup je mrežnog prometa označenog prvim pravilom. Ukoliko prvo pravilo nema prekidajuću akciju, tada ono mora imati identičnu akciju (i akcijske parametre) kao i drugo pravilo. Osim toga, između pravila se ne smiju pronaći druga pravila s neprekidajućom akcijom i akcijskim parametrima zbog mogućnosti da takva pravila modificiraju paket, koji potom pak može biti procesuiran preko „zasjenjenog“ pravila te se stoga to pravilo ne smije uklanjati.

Kako bi se jedno sigurnosno pravilo našlo „u sjeni iza“ nekoga drugog sigurnosnog pravila to pravilo mora imati prekidajuću akciju (ili jednaku akciju i akcijske parametre kao i drugo pravilo), a pri tome moraju biti zadovoljene Definicije 2 ili 3 iz poglavlja 2.2. Svi parametri usporedbe u navedena dva pravila moraju biti identični ili svi parametri drugog „zasjenjenog“ pravila moraju biti podskup pripadnih parametara prvog pravila.

Utvrđivanje je li jedno pravilo uključeno u drugo pravilo zasniva se na analizi uključenosti njihovih parametara usporedbe (Poglavlje 6.6.2).

Primjer uključenosti drugog pravila u prvo pravilo:

```
R1:
-A FORWARD -s 2.4.5.0/24 -d 1.2.0.0/16 -i eth+ ! -o
eth0 -p tcp -m tcp --sport 1024:65535 ! --dport 80:124
! --tcp-option 11 -m dscp ! --dscp 0x00 -m ttl --ttl-gt
20 -m conntrack --ctstate NEW,RELATED,ESTABLISHED -m
mark ! --mark 0x15 -m mac ! --mac-source
AA:00:33:44:55:66 -m pkttype ! --pkt-type multicast -m
length --length 100:1000 -m tcpmss --mss 0:2000 -m time
--datestart 2009-07-01 --datestop 2012-01-01T12:00:00 -
-timestart 08:00 -j DROP

R2:
-A FORWARD -s 2.4.5.0/26 -i eth0 -o eth1 -p tcp -m tcp
--sport 1024:10000 --tcp-option 12 -m dscp --dscp 0x20
-m iprange --dst-range 1.2.3.5-1.2.7.9 -m ttl --ttl-eq
40 -m conntrack --ctstate NEW,RELATED -m mark --mark
0x22 -m mac --mac-source 11:22:33:44:55:66 -m tos --tos
0x10 -m pkttype --pkt-type unicast -m length --length
200:900 -m tcpmss ! --mss 1000:65535 -m multiport --
dports 21,22,443 -m time --datestart 2011-01-01 --
datestop 2011-03-01 --timestart 12:00 --timestop 16:00
-j REJECT --reject-with icmp-net-unreachable
```

Kako bi se neko pravilo (R2) nalazilo u „sjeni“ drugog pravila (R1) svaki parametar iz R2 mora sadržajno biti podskup ili jednak pripadajućem parametru iz R1. U navedenom primjeru to je slučaj jer:

- izvorna IP mreža `2.4.5.0/26` iz R2 je podskup IP mreže `2.4.5.0/24` iz R1,
- ciljni IP raspon `1.2.3.5-1.2.7.9` iz R2 je podskup IP mreže `1.2.0.0/16` (1.2.0.0-1.2.255.255) iz R1,
- ulazno sučelje `eth0` iz R2 je podskup svih Eth ulaznih sučelja `eth+` iz R1,
- izlazno sučelje `eth1` je jednako u oba pravila,
- protokol `tcp` je jednak u oba pravila,

- izvorni portovi `1024:10000` iz R2 podskup su mogućih izvornih portova `1024:65535` iz R1,
- ciljni portovi `21,22,443` iz R2 podskup su mogućih ciljnih portova koji nisu u rasponu `80:124` (`! 80:124`) iz R1,
- TCP opcija `12` iz R2 je podskup svih mogućih TCP opcija iz R1 različitih od `11` (`! 11`), što automatski uključuje i `12`,
- DSCP vrijednost `0x20` iz R2 je podskup svih mogućih TCP opcija iz R1 različitih od `0x00` (`! 0x00`), što automatski uključuje i `0x20`,
- TTL vrijednost `250` iz R2 je podskup svih mogućih TTL vrijednosti iz R1 većih od `20` (`gt 20 – greater than 20`), što automatski uključuje i `250`,
- stanja konekcije `NEW,RELATED` iz R2 su podskup stanja konekcija `NEW,RELATED,ESTABLISHED` iz R1, jer je u R1 dozvoljeno i `ESTABLISHED` stanje,
- *mark* oznaka `0x22` iz R2 je podskup svih mogućih *mark* oznaka iz R1 različitih od `0x15` (`! 0x15`), što automatski uključuje i `0x22`,
- izvorna MAC adresa `11:22:33:44:55:66` iz R2 je podskup svih mogućih MAC adresa iz R1 različitih od `AA:00:33:44:55:66` (`!AA:00:33:44:55:66`), što automatski uključuje i `11:22:33:44:55:66`,
- ToS vrijednost iz R2 je podskup svih mogućih ToS vrijednosti iz R1 (u R1 nije navedena niti jedna ToS vrijednost što automatski uključuje sve moguće) ,
- tip mrežnih paketa `unicast` iz R2 je podskup svih mogućih tipova mrežnih paketa iz R1 različitih od `multicast` (`! multicast`), što automatski uključuje i `unicast`,
- duljina mrežnog paketa `200:900` iz R2 je podskup duljine mrežnih paketa `100:1000` iz R1,
- maksimalna veličina TCP segmenta MSS koja ima maksimalnu veličinu 999 (`! 1000:65535`) iz R2 je podskup MSS vrijednosti `0:2000` iz R1,
- datumski interval `2011-01-01` do `2011-03-01` iz R2 podskup je datumskog intervala od `2009-07-01` do `2012-01-01T12:00:00` iz R1,
- vremenski interval `12:00` do `16:00` iz R2 podskup je vremenskog intervala od `08:00` do kraja dana iz R1.

Ako je akcija prvog pravila prekidajuća, akcija pravila „u sjeni iza“ za proces uklanjanja anomalije nije važna jer se to pravilo nikad ne ispunjava. Tako ta akcija može biti prekidajuća (prekida se procesuiranje) ili neprekidajuća (nakon što se paket uspješno prepozna nastavlja se procesuiranje ostalih pravila). A ako akcija prvog pravila nije prekidajuća tada drugo pravilo mora imati istu akciju i iste akcijske parametre kako se ne bi ista akcija (npr. logiranja) nepotrebno ponavljala.

Za prethodni primjer namjerno su odabrana sigurnosna pravila s velikim brojem parametara različitog oblika kako bi se istaknula složenost identificiranja zavisnosti jer to na prvi pogled nije nikako jasno vidljivo, osobito ukoliko se pravila nalaze u listi većeg broja pravila, pri čemu se između njih nalazi niz drugih pravila. Iako nije uobičajeno, pravila teoretski mogu biti i složenija jer nisu svi mogući parametri uključeni.

6.1.4. Uklanjanje anomalije u „sjeni ispred“

Anomalija u „sjeni ispred“ (engl. *Shadowed before*) već je definirana u poglavlju 3.1 kao kombinacija sigurnosnih pravila gdje drugo pravilo (u definiranom poretku) označava sav mrežni promet koji označava i prvo pravilo. Za razliku od prethodne opisane anomalije gdje akcije nisu morale biti uvijek identične, kod anomalije u „sjeni ispred“ akcije i akcijski parametri tih dvaju pravila moraju biti identični, a između njih ne smiju se pronaći druga pravila s kojima se ta dva pravila nalaze u relaciji; osim u slučaju da ta među-pravila imaju identičnu akciju i akcijske parametre. Dodatno, ako prvo pravilo ima neprekidajuću akciju i barem jedan akcijski parametar, između njih se ne smiju nalaziti nikakva pravila koja imaju neprekidajuću akciju i akcijske parametre. Ali ako dva promatrana pravila imaju neprekidajuću akciju i nemaju akcijskih parametara tada se između njih smiju pronaći pravila s kojima se nalaze u relaciji i koja imaju različitu neprekidajuću akciju te nemaju akcijske parametre.

Procedura utvrđivanja je li jedno pravilo uključeno u drugo pravilo ista je kao i kod anomalije u „sjeni iza“ (Poglavlje 6.1.3), a zasniva se na analizi uključenosti njihovih parametara usporedbe (Poglavlje 6.6.2). Za razliku od anomalije u „sjeni iza“ gdje parametri iz drugog pravila moraju biti sadržani ili jednaki parametrima prvog pravila, kod anomalije u „sjeni ispred“ nužna je obrnuta situacija tj. svaki parametar iz prvog pravila (R1) mora sadržajno biti podskup ili jednak pripadajućem parametru iz drugog pravila (R2).

Kako bi utvrdio nalaze li se između dva promatrana pravila druga pravila s prekidajućim akcijama s kojima se oni nalaze u relaciji, Firo koristi direktne acikličke grafove - DAG (Poglavlje 4.3).

Primjer anomalije u „sjeni ispred“:

```
R1:
-A FORWARD -s 192.168.10.15/32 -p tcp -m recent
  --update --seconds 100 --name ABC --rsource -j ACCEPT

Rx:
-A FORWARD -p ALL -s 192.168.11.0/24 -j DROP

R2:
-A FORWARD -s 192.168.10.0/24 -m recent --update
```

```
   --seconds 100 --name ABC --rsource -j ACCEPT
```

U navedenom primjeru dozvoljeno je uklanjanje pravila R1 zbog anomalije u „sjeni ispred“. Naime, svi uspoređujući parametri od R1 su uključeni u parametre od R2:

- izvorna IP adresa `192.168.10.15` iz R1 je podskup IP mreže `192.168.10.0/24` iz R2,
- protokol `tcp` iz R1 je podskup svih protokola iz R2,
- *recent* naziv liste je isti `ABC` u oba pravila,
- Broj sekundi unutar kojih je morala biti zabilježena prethodna IP adresa je isti u oba pravila – `100.`

Uklanjanje je dozvoljeno i jer su akcija i akcijski parametri isti:

- akcija u oba pravila je ista (ACCEPT),
- u oba pravila je naveden akcijski parametar `--update` koji mijenja *timestamp* zapis za pronađenu IP adresu,
- u oba pravila je definirano uspoređivanje i spremanje izvornih IP adresa (akcijski parametar `--rsource`).

Iako se između dva pravila nalazi drugo pravilo (Rx), ono ne ometa uklanjanje pravila u „sjeni ispred“ jer:

- nije u relaciji s promatranim pravilima zbog različitog skupa izvornih IP adresa (`192.168.11.0/24`) koji se ne preklapa ni s izvornim IP adresama iz pravila R1 (`192.168.10.15`), ni iz R2 (`192.168.10.0/24`).

6.1.5. Uklanjanje zadnjih nepotrebnih pravila

Uklanjanje zadnjih nepotrebnih pravila slično je prethodno opisanom optimiranju uklanjanja anomalija u „sjeni ispred“. Razlika je u tome što se u ovom slučaju ne uspoređuju dva pravila, već se sva pravila koja imaju akciju identičnu osnovnoj akciji sigurnosne politike (ACCEPT ili DROP) te nemaju ikakvih drugih akcijskih parametara, uklanjaju pod uvjetom da se iza njih ne nalaze nikakva druga „prekidajuća“ pravila s kojima se oni nalaze u relaciji.

Kada bi sva sigurnosna pravila u nekoj listi imala istu akciju kao osnovna politika tog lanca tada bi se sva pravila mogla ukloniti.

Primjer uklanjanja zadnjih pravila s istom akcijom kao što je i akcija lanca:

```
:FORWARD ACCEPT [0:0]

R1:
-A FORWARD -s 1.1.1.1/32 -j ACCEPT
```

```
R2:
-A FORWARD -s 10.10.10.0/24 -j ACCEPT

R3:
-A FORWARD -s 2.2.2.2/32 -p tcp -m recent --update --
seconds 100 --name ABC --rsource -j ACCEPT

R4:
-A FORWARD -p ALL -s 3.3.3.3/32 -j DROP

R5:
-A FORWARD -p ALL -s 10.10.0.0/16 -j DROP
```

Među navedenim pravilima jedino pravilo R1 se može ukloniti. Ono ima istu akciju koja je definirana i za FORWARD lanac (ACCEPT), a ne nalazi se u relaciji s niti jednim narednim pravilom (R2-R5) zbog izvorne IP adrese koja se ne preklapa ni s jednom izvornom adresom ili mrežom narednih pravila.

6.1.6. Spajanje sigurnosnih pravila

Jedan od koraka u optimiranju je i spajanje dva pravila. Preduvjeti za dozvoljeno spajanje dvaju pravila su sljedeći:

- akcije tih pravila moraju biti identične,
- akcijski parametri moraju biti identični,
- između tih dvaju pravila se ne smiju naći druga pravila koja su s njima u relaciji, a da imaju različitu akciju ili akcijske parametre,
- između pravila se ne smiju naći druga pravila koja imaju neprekidajuću akciju i akcijske parametre,
- parametri uspoređivanja moraju biti identični – osim jednog različitog:
    - različit parametar uspoređivanja dozvoljava spajanje.

Spajanje dvaju pravila najsloženiji je mehanizam Firo optimizatora i postoje različite metode spajanja za različite tipove parametara kao što je opisano u poglavlju 6.6.3.

Primjer spajanja sigurnosnih pravila po IP adresi:

```
MERGE RULES:
   [R1] -A INPUT -s 1.1.1.0/25 -j LOG
   [R2] -A INPUT ! -s 1.1.1.0/24 -j LOG
  [New] –A INPUT ! -s 1.1.1.128/25 -j LOG
```

U navedenom primjeru prvo pravilo R1 uključuje mrežu `1.1.1.0/25` (`1.1.1.0-1.1.1.127`), drugo pravilo R2 uključuje sve IP adrese koje nisu iz mreže `1.1.1.0/24` (`1.1.1.0-1.1.1.255`). Sumarno, ta dva pravila

uključuju sve IP adrese koja nisu iz mreže `1.1.1.128/25` (`1.1.1.128-1.1.1.255`).

Primjer spajanja sigurnosnih pravila po protokolu, ali i znakovnim nizovima koji nemaju niz mogućih vrijednosti:

```
MERGE RULES:
   [R1] -A FORWARD -d 2.2.2.2 –p tcp -j DROP
   [R1] -A FORWARD -d 2.2.2.2 ! –p tcp -j DROP
  [New] –A FORWARD -d 2.2.2.2 –p ALL -j DROP
```

U navedenom primjeru prvo pravilo uključuje pakete s TCP protokolom, drugo pravilo R2 uključuje sve pakete koji imaju protokol različit od TCP-a. Sumarno, ta dva pravila uključuju sve protokole.

Primjer spajanja sigurnosnih pravila po znakovnom nizu:

```
MERGE RULES:
[R1]   -A INPUT -d 3.3.3.3 -m state --state ESTABLISHED
  -j REJECT
[R2]   -A INPUT -d 3.3.3.3 -m state --state RELATED
  -j REJECT
[New]  -A INPUT -d 3.3.3.3 -m state --state
  ESTABLISHED,RELATED -j REJECT
```

U navedenom primjeru prvo pravilo R1 uključuje pakete iz uspostavljenih konekcija (`ESTABLISHED`), drugo pravilo R2 uključuje pakete iz povezanih konekcija (`RELATED`). Sumarno, ta dva pravila uključuju pakete iz uspostavljenih i povezanih konekcija (`ESTABLISHED,RELATED`).

Primjer spajanja sigurnosnih pravila po portovima (brojevima):

```
MERGE RULES:
[1] -A POSTROUTING -s 4.4.4.4 --dport 0:1024
  -j MASQUERADE
[2] -A POSTROUTING -s 4.4.4.4 --dport 5000:65535
  -j MASQUERADE
[New] -A POSTROUTING -s 4.4.4.4 ! --dport 1025:4999
  -j MASQUERADE
```

U navedenom primjeru prvo pravilo R1 uključuje pakete usmjerene prema servisima na portovima `0:1024`, drugo pravilo R2 uključuje pakete usmjerene prema servisima na portovima `5000:65535`. Sumarno, ta dva pravila uključuju pakete koji nisu usmjereni prema servisima na portovima `1025:4999`.

Primjer spajanja sigurnosnih pravila po datumu:

```
MERGE RULES:
[R1] -A INPUT -s 5.5.5.5 -m time --datestart 2010-07-01
  --datestop 2011-07-01 -j DROP
[R2] -A INPUT -s 5.5.5.5 -m time --datestart 2010-12-01
  --datestop 2012-07-01 -j DROP
[New]-A INPUT -s 5.5.5.5 -m time --datestart 2010-07-01
  --datestop 2012-07-01 -j DROP
```

U navedenom primjeru prvo pravilo R1 odbacuje pakete s IP adrese `5.5.5.5` pristigle između `2010-07-01` i `2011-07-01`. Drugo pravilo R2 odbacuje pakete s IP adrese `5.5.5.5` pristigle između `2010-12-01` i `2012-07-01`. Sumarno, ta dva pravila odbacuju pakete s IP adrese `5.5.5.5` pristigle između `2010-07-01` i `2012-07-01`.

Kako bi se u svim navedenim primjerima spojila navedena pravila između njih ne smiju se nalaziti ostala pravila s kojima se drugo promatrano pravilo nalazi u relaciji. Prvo pravilo može biti u relaciji s tim među-pravilima, ali drugo ne smije jer se drugo pravilo spaja s prvim – novo pravilo zauzima poziciju prvog pravila.

6.1.7. Uklanjanje redundantnih parametara

Osim uklanjanja pravila, Firo uklanja i nepotrebne parametre iz pravila. Često pravila imaju određene parametre koji zahtijevaju dodatno procesuiranje, iako je postojanje takvih parametara nepotrebno zbog drugih sigurnosnih pravila koja prethode analiziranom pravilu. Naime, parametar u analiziranom pravilu zajedno s parametrom iz prethodnog pravila pokriva sve moguće vrijednosti, stoga je parametar iz drugog pravila moguće ukloniti. Preduvjet za uklanjanje parametra iz drugog pravila je akcija prvog pravila koja mora biti prekidajuća, a svi ostali parametri uspoređivanja moraju biti identični. Akcija drugog pravila može biti bilo kojeg tipa. U praksi, zbog optimiranja spajanjem pravila te dvije akcije i njihovi akcijski parametri nikada ne mogu biti identični jer bi se inače ta dva pravila spojila. Također, prije drugog pravila mogu se pronaći i među-pravila s kojima se ono nalazi u relaciji, ali ta među-pravila moraju biti ili prekidajuća ili neprekidajuća bez akcijskih parametara koji bi mogli izmijeniti paket, konekciju ili druge podatke koje pravila provjeravaju.

Primjer uklanjanja parametra TTL iz sigurnosnog pravila:

```
REMOVED PARAMETER (ttl) FROM RULE:
[R_Other]
  -A INPUT -s 1.4.0.0/24 -m ttl --ttl-eq 231 –j ACCEPT
[R_Old]
  -A INPUT -s 1.4.0.0/24 -m ttl ! --ttl-eq 231 -j DROP
[R_New]
  -A INPUT -s 1.4.0.0/24 -j DROP
```

U prethodnom primjeru prvo pravilo R_{Other} prihvaća sve pakete s mreže `1.4.0.0/24` koji imaju TTL vrijednost 231. Drugo pravilo R_{Old} odbacuje sve pakete s mreže `1.4.0.0/24` koji imaju TTL vrijednost različitu od 231. Iz tog drugog pravila može se ukloniti usporedba po TTL vrijednosti jer će zbog prvog pravila R_{Other} mrežni paketi koji dolaze s mreže `1.4.0.0/24` i koji se uspoređuju nakon pravila R_{Other} uvijek u tom koraku imati TTL različit od 231. Stoga se pravilo R_{Old} može zamijeniti s pravilom R_{New}.

Primjer uklanjanja znakovnog parametra iz sigurnosnog pravila:

```
REMOVED PARAMETER (state) FROM RULE:
[R_Other]
  -A INPUT -p tcp -s 4.4.4.4 -m state
    --state INVALID,NEW,UNTRACKED -j DROP
[R_Old]
  -A INPUT -p tcp -s 4.4.4.4 -m state
    --state ESTABLISHED,RELATED -j ACCEPT
[R_New]
  -A INPUT -p tcp -s 4.4.4.4 -j ACCEPT
```

U prethodnom primjeru prvo pravilo R_{Other} odbacuje sve TCP pakete s IP adrese `4.4.4.4` koji su dio nove, invalidne ili nepraćene konekcije (`INVALID,NEW,UNTRACKED`). Drugo pravilo R_{Old} prihvaća sve TCP pakete s IP adrese `4.4.4.4` koji su dio uspostavljene ili povezane konekcije (`ESTABLISHED,RELATED`). Budući da se skup mogućih stanja sastoji od 5 navedenih stanja, iz tog drugog pravila može se ukloniti usporedba po stanju konekcije jer će zbog prvog pravila R_{Other} mrežni TCP paketi koji dolaze s računala `4.4.4.4` i koji se uspoređuju nakon pravila R_{Other} uvijek biti dio uspostavljene ili povezane konekcije (`ESTABLISHED,RELATED`). Zato se pravilo R_{Old} zamjenjuje s pravilom R_{New}.

Primjer uklanjanja IP adrese iz sigurnosnog pravila:

```
REMOVED PARAMETER (destination_ip) FROM RULE:
[R_Other]
 -A FORWARD –i eth0 ! -d 1.16.16.7 -j ACCEPT
[R_Old]
 -A FORWARD –i eth0 -d 1.0.0.0/255.0.255.0 -j DROP
[R_New]
 -A FORWARD –i eth0 -j DROP
```

U prethodnom primjeru prvo pravilo R_{Other} prihvaća sve pakete s mrežnog sučelja `eth0` koji nisu usmjereni prema IP adresi `1.16.16.7`. Drugo pravilo R_{Old} odbacuje sve pakete s mrežnog sučelja `eth0` koji su usmjereni prema IP adresama `1.*.16.*`, što uključuje i IP adresu `1.16.16.7`. Dakle, parametri za ciljnu adresu u pravilima R_{Other} i R_{Old} čine skup svih mogućih IP adresa te se pravilo R_{Old} zamjenjuje s pravilom R_{New}.

Primjer uklanjanja portova (brojeva) iz sigurnosnog pravila:

```
REMOVED PARAMETER (destination_port) FROM RULE:
[R_Other]
  -A OUTPUT -d 1.1.1.1 -m multiport
    ! --dports 3,4,5,6,8,11,17 -j ACCEPT
[R_Old]
  -A OUTPUT -d 1.1.1.1 --dport 3:50 -j DROP
[R_New]
  -A OUTPUT -d 1.1.1.1 -j DROP
```

U prethodnom primjeru prvo pravilo R_{Other} prihvaća sve pakete usmjerene prema računalu s IP adresom `1.1.1.1` i portovima različitima od `3,4,5,6,8,11,17`. Drugo pravilo R_{Old} odbacuje sve pakete usmjerene prema računalu s IP adresom `1.1.1.1` i portovima u rasponu `3:50`. Parametri za ciljne portove u pravilima R_{Other} i R_{Old} čine skup svih mogućih portova (`0:65535`) te se pravilo R_{Old} može zamijeniti pravilom R_{New}.

6.1.8. Uklanjanje redundantnih elemenata iz parametara

Uklanjanje redundantnih elemenata iz parametara trenutno je usmjereno samo na uklanjanje portova iz niza portova, kao i na uklanjanje elemenata iz niza znakovnih vrijednosti. Uklanjanje se obavlja jer ti elementi već postoje u nekom prethodnom pravilu. Time pravila postaju manje kompleksna, a ponekad se dobiveni niz portova može konvertirati u jednostavniji raspon ili omogućiti optimiranje pravila pomoću drugih metoda optimiranja.

U pravilu, prva akcija među promatranim pravilima treba biti prekidajuća, pri čemu se između dvaju promatranih pravila mogu pronaći i neka druga pravila s kojima se oni nalaze u relaciji. Ukoliko prva akcija nije prekidajuća, akcije promatranih pravila moraju biti identične te prvo pravilo ne smije imati definirane akcijske parametre. Ali tada među-pravila (između prvog i drugog pravila) moraju biti ili prekidajuća ili neprekidajuća bez akcijskih parametara. Kod neprekidajućih akcija ne smiju postojati akcijski parametri jer oni mogu modificirati paket, konekciju ili druge promatrane elemente.

Za razliku od prethodne optimizacijske tehnike, kod uklanjanja redundantnih elemenata iz parametara prva akcija može biti i neprekidajuća. To je dozvoljeno jer se uklanjanjem elemenata smanjuje broj paketa na koje će se pravilo primijeniti. Dok se kod uklanjanja cijelog parametra povećava broj paketa na koje će se pravilo primijeniti. Dakle, kod ove optimizacijske tehnike ne postoji opasnost npr. duplog logiranja.

```
REMOVED REDUNDANT ELEMENTS FROM RULE:
[R_Other]
-A INPUT -s 5.5.5.5 -m multiport --dports  90,91,92,93,
```

```
171,172,173,176,178,211,212,213,214,215,333 -j ACCEPT
 [ROld]
-A INPUT -s 5.5.5.5 -m multiport –-dports 171,172,210,
211,212,213,214,215,216,217,218,219,220,221,225       -j
ACCEPT
 [RNew]
-A INPUT -s 5.5.5.5 -m multiport -–dports
  210,216,217,218,219,220,221,225 -j ACCEPT
```

U primjeru prvo pravilo R_{Other} dozvoljava IP pakete s izvora `5.5.5.5` usmjerene na 15 portova. Drugo pravilo R_{Old} dozvoljava pakete s izvora `5.5.5.5` usmjerene na 15 portova od čega je 7 portova (`171,172,211,212,213,214,215`) navedeno i u prvom pravilu R_{Other} (akcija je prekidajuća – paketi se prihvaćaju). Kad bi na Iptables alatu bilo dozvoljeno imati više od 15 portova u jednom parametru ta bi se dva pravila mogla spojiti jer im je akcija ista. Ali zbog navedenog ograničenja spajanje nije dozvoljeno pa se iz drugog pravila R_{Old} može ukloniti 7 navedenih portova i dobije se R_{New}.

Uklanjanje redundantnih elemenata iz parametara obavlja se i za niz znakovnih vrijednosti kao što se vidi u sljedećem primjeru:

```
REMOVED REDUNDANT ELEMENTS FROM RULE:
[ROther]
-A   FORWARD   -d   1.1.1.1   -m   addrtype   --src-type
LOCAL,MULTICAST,NAT,PROHIBIT,THROW -j LOG
 [ROld]
-A   FORWARD   -d   1.1.1.1   -m   addrtype   !   --src-type
BLACKHOLE,MULTICAST,NAT,UNICAST,UNREACHABLE -j LOG
 [RNew]
-A FORWARD -d 1.1.1.1 -m addrtype --src-type ANYCAST,
BROADCAST,UNSPEC,XRESOLVE -j LOG
```

U primjeru prvo pravilo R_{Other} logira pakete usmjerene prema računalu s `1.1.1.1` IP adresom, pri čemu je tip izvorne adrese određenog tipa (`LOCAL,MULTICAST,NAT,PROHIBIT,THROW`). Drugo pravilo R_{Old} logira pakete usmjerene prema računalu s `1.1.1.1` IP adresom, pri čemu tip izvorne adrese ne smije biti određenoga tipa (`BLACKHOLE,MULTICAST,NAT,UNICAST,UNREACHABLE`). Budući da ne smije biti tog navedenog tipa, a u prethodnom pravilu su već logirani drugi izvorni tipovi (`LOCAL,MULTICAST,NAT,PROHIBIT,THROW`), za drugo pravilo može se specificirati kako treba logirati samo preostale tipove (`ANYCAST, BROADCAST,UNSPEC,XRESOLVE`). I zbog toga se pravilo R_{Old} može zamijeniti s R_{New}.

6.1.9. Optimiranje LOG pravila

Korištenje LOG pravila na optimiranim vatrozidima nije idealno jer LOG pravila zahtijevaju kreiranje *log* (dnevničkih) zapisa koji se moraju zapisivati na disk ili slati na udaljeni poslužitelj, a to utječe na brzinu obrade i na performanse vatrozida. Ipak, ta pravila su često potrebna, osobito za naknadne analize pa je i na njih potrebno primijeniti optimizacijske tehnike.

Osim što se na LOG pravila primjenjuju sve optimizacijske metode (osim metode uklanjanja redundantnih parametara) opisane u prethodnim poglavljima, na njih se primjenjuje i metoda premještanja LOG pravila neposredno ispred relacijskih pravila. Naime, nakon što se provedu sve optimizacijske procedure te više nema redundantnih pravila za optimizirati, LOG pravila se premještaju na kasnije pozicije u nizu sigurnosnih pravila kako bi se pozicionirala neposredno ispred pravila koja će nad tim „logiranim“ mrežnim pravilima provesti određenu akciju (prihvatiti, odbaciti ili uraditi neku drugu specifičnu akciju). Također, pri premještanju se ne mogu pozicionirati poslije pravila s neprekidajućim akcijama i akcijskim parametrima koje bi mogle promijeniti bitne elemente.

Premještanje LOG pravila zadnji je korak optimizacijske procedure. LOG pravila premještaju se neposredno ispred onih pravila koja nisu LOG pravila, a s kojima se nalaze u relaciji.

Primjer premještanja LOG pravila:

```
## RULES
[R1] -A FORWARD -p tcp -s 1.2.1.0/26 -j ULOG
         --ulog-nlgroup 23
[R2] -A FORWARD -p tcp -s 1.1.1.0/24 -j DROP
[R3] -A FORWARD -p tcp -s 1.1.7.0/24 -j DROP
[R4] -A FORWARD -p tcp -s 1.2.1.0/24 -j DROP

## REPOSITION OF LOG RULES
[R2] -A FORWARD -p tcp -s 1.1.1.0/24 -j DROP
[R3] -A FORWARD -p tcp -s 1.1.7.0/24 -j DROP
[R1] -A FORWARD -p tcp -s 1.2.1.0/26 -j ULOG
         --ulog-nlgroup 23
[R4] –A FORWARD -p tcp -s 1.2.1.0/24 -j DROP
```

U navedenom primjeru prvo LOG R1 pravilo „preskače“ drugo R2 i treće R3 pravilo s kojima se ne nalazi u relaciji te se pozicionira ispred četvrtog R4 pravila s kojim se nalazi u relaciji.

6.1.10. Optimiranje pravila s opcijom limitiranja prometa

Jedan od parametara u sigurnosnim pravilima je opcija limitiranja mrežnih paketa. Broj dozvoljenih paketa određuje se po jedinici vremena (sekunde, minute, sati, dani), a svrha je ograničiti broj logiranih mrežnih paketa ili prihvaćenih mrežnih paketa, npr. kako bi se onemogućio DoS (engl. *Denial of Service*) napad. Opcija limitiranja ne može se smatrati parametrom uspoređivanja, ali niti akcijskim parametrom. Za takva pravila Firo dozvoljava korisniku konfiguriranje treba li dozvoliti:

- uklanjanje anomalija u „sjeni ispred“ u slučaju kad je opcija limitiranja identična,
- uklanjanje anomalija u „sjeni iza“ u slučaju kad je opcija limitiranja identična,
- spajanje sigurnosnih pravila kad je opcija limitiranja identična,
- spajanje sigurnosnih pravila kad su svi parametri identični osim opcije limitiranja.

Prve tri opcije već su opisane u prošlim poglavljima (6.1.3, 6.1.4, 6.1.6) dok četvrta opcija predstavlja poseban slučaj spajanja parametra, za što svi parametri osim *limit* parametra moraju biti identični, a ograničenje u oba pravila mora biti izraženo u istoj vremenskoj jedinici (sekunda, minuta, sat, dan). Primjer spajanja pravila po parametru limitiranja:

```
MERGE RULES:
[R1]  -A INPUT -s 1.1.1.0/24 -m limit --limit 20/minute
  -j ACCEPT
[R2]  -A INPUT -s 1.1.1.0/24 -m limit --limit 10/minute
  -j ACCEPT
[New] -A INPUT -s 1.1.1.0/24 -m limit --limit 30/minute
  -j ACCEPT
```

U navedenom primjeru prvo pravilo R1 prihvaća maksimalno 20 paketa po minuti R1 pristiglih s mreže `1.1.1.0/24`. Drugo pravilo R2 prihvaća maksimalno 10 paketa po minuti R1 pristiglih s mreže `1.1.1.0/24`. Sumarno, ta dva pravila prihvaćaju maksimalno 30 paketa po minuti R1 pristiglih s mreže `1.1.1.0/24`.

6.2. Konfiguracija programa

Firo nakon pokretanja analizira `config.txt` datoteku koja se treba nalaziti u istom direktoriju kao i program.

Parametri `firewall_directory` i `firewall_file` određuju gdje se nalazi i kako se zove datoteka u kojoj se nalazi niz sigurnosnih pravila koje je

potrebno optimizirati. Ako direktorij nije naveden datoteka se traži u trenutnom direktoriju, u kojem je Firo pokrenut.

```
firewall_directory=C:\Firewall
firewall_file =IPTablesINPUT.txt
```

Parametar `report_directory` određuje gdje se spremaju optimirane verzije sigurnosnih pravila, dok parametar `report_prefix` određuje prefiks generiranih datoteka. Svaki put kad se obavi određeni tip optimizacije generira se nova datoteka s optimiranim pravilima (npr. `vatrozid_0.txt`, `vatrozid_1.txt`,`vatrozid_2.txt`, ...). Na kraju se generira i datoteka s opisom svih optimizacija (`vatrozid_log_changes.txt`).

```
report_directory = C:\Firewall\Output
report_prefix = vatrozid
```

Parametar `log_directory` određuje u koji direktorij se spremaju dnevnički zapisi (engl. *log*) koji opisuju detalje o izvođenju Firo programa. Parametar `debug_file` određuje datoteku u koju se spremaju dnevnički zapisi (engl. *log*) o izvođenju Firo programa. Parametar `debug_level` određuje koliko se detaljno trebaju spremati informacije (`NONE`, `INFO`, `ALL`), dok parametar `debug_output` određuje trebaju li se detalji prikazivati tijekom rada i/ili spremati u datoteku.

```
log_directory = C:\Firewall\Log
debug_file = firewall_debug.log
debug_level = ALL
debug_output = monitor
debug_output = file
```

Sljedeći parametar `os` određuje na kojem se sustavu pokreće Firo program (`windows`, `linux`):

```
os = windows
```

Predefinirano se svi koraci optimizacije izvode, ali ih je preko konfiguracijske datoteke moguće isključiti. Sljedećih 7 parametara definira koje korake optimizacije ne treba provoditi (predefinirano su omogućeni):

```
remove_shadowed_before_rules = no
remove_shadowed_after_rules = no
remove_last_rules = no
merge_rules = no
remove_redundant_parameters = no
remove_redundant_array_elements = no
reposition_log_rules = no
```

Za potrebe optimiranja pravila s opcijom limitiranja Firo omogućava konfiguriranje treba li dozvoliti uklanjanje anomalija u „sjeni ispred“ u slučaju kad je opcija limitiranja identična (`remove_limit_rules_shadowed_before`), uklanjanje anomalija u „sjeni iza“ u slučaju kad je opcija limitiranja identična (`remove_limit_rules_shadowed_after`), spajanje sigurnosnih pravila kad je opcija limitiranja identična (`merge_limit_rules_with_equal_limit`), spajanje sigurnosnih pravila kad su svi parametri identični osim opcije limitiranja (`merge_limit_rules_by_limit_parameter`).

```
remove_limit_rules_shadowed_before = no
remove_limit_rules_shadowed_after = no
merge_limit_rules_with_equal_limit = no
merge_limit_rules_by_limit_parameter = yes
```

Budući da se u datoteci sa sigurnosnim pravilima mogu pronaći i neke komponente koje ne treba ili nije moguće optimizirati, njihovo ignoriranje moguće je realizirati navođenjem `ignore_chain` i `ignore_element`. Tu se misli na lance naredbi koje ne treba optimizirati, a u slučaju Iptables alata oznake tablica koje su dio izlaza `save-iptables` naredbe.

```
ignore_chain = allowed
ignore_chain = bad_tcp_packets
ignore_chain = ISPITAJ_TCP

ignore_element = COMMIT
ignore_element = *nat
ignore_element = *filter
ignore_element = *mangle
ignore_element = *raw
```

U konfiguracijskoj datoteci određuju se i oznake prihvaćanja (`accept_rule`), odbacivanja (`reject_rule`) i logiranja (`log_rule`) mrežnih podataka:

```
accept_rule = -j ACCEPT
accept_rule = ACCEPT

reject_rule = -j DROP
reject_rule = DROP

log_rule = -j LOG
log_rule = -j ULOG
log_rule = -j NFLOG
```

Kako bi se odredili lanci sigurnosnih pravila koje je potrebno optimizirati koristi se `optimize_chain`:

```
optimize_chain = INPUT
optimize_chain = OUTPUT
optimize_chain = FORWARD
```

Budući da Firo omogućuje optimiranje sigurnosnih pravila korištenjem različitih akcija (ne samo akcije prihvaćanja i odbacivanja), potrebno je odrediti koje akcije su „prekidajuće“ – prekidaju procesuiranje mrežnog paketa u lancu naredbi, a za to se koristi `breaking_action` parametar:

```
breaking_action = REJECT
breaking_action = DNAT
breaking_action = SNAT
breaking_action = MASQUERADE
```

Jedna od posebnosti Iptables alata je korištenje niza portova pri čemu je maksimalan broj dozvoljenih portova 15. U slučaju kada se to ograničenje promijeni dovoljno je promijeniti `port_array_size_max` parametar. A kad se spaja raspon portova i raspon portova ili raspon portova i jedan port, definirani rasponi moraju se raspisati kao niz portova. U slučaju da administrator ne želi takvu konverziju, dovoljno je definirati optimalni broj portova u nizu (`port_array_size_optimal`) koji se ne može prijeći konverzijom raspona portova u niz portova:

```
port_array_size_optimal = 10
port_array_size_max = 15
```

Dodatno, Iptables alat ima i ograničenje jer ne dozvoljava kombiniranje izvornog i ciljnog porta ukoliko je jedan od njih niz portova. Ako se to dogodi Iptables ignorira drugi navedeni parametar. Međutim, ukoliko to iz nekog razloga nije potrebno, nužno je postaviti sljedeći parametar kako bi se ignoriralo navedeno ograničenje:

```
multiport_ignore_2nd = yes
```

Trenutno, konfiguracija Firo alata se ne obavlja samo kroz `config.txt` konfiguracijsku datoteku. Određivanje tipova parametara obavlja se kroz `Configuration.cpp` programsku klasu, a ukoliko nisu navedeni, svi pronađeni nedefinirani parametri u sigurnosnim pravilima smatraju se znakovnim nizom (*string*).

6.3. Ulazni podaci

Kao ulaz za Firo optimizator koristi se rezultat `iptables-save` naredbe. Primjer dohvaćanja liste Iptables pravila definirane na Linux sustavu[3]:

```
[root@test ~]# iptables-save > lista_pravila.txt
```

Iz čega se dobije sljedeći sadržaj:

```
# Generated by iptables-save v1.4.10 on Mon Mar  7 06:12:28 2011
*raw
:PREROUTING ACCEPT [23015817:2831083956]
:OUTPUT ACCEPT [0:0]
-A OUTPUT -j ACCEPT
COMMIT
# Completed on Mon Mar  7 06:12:28 2011

# Generated by iptables-save v1.4.10 on Mon Mar  7 06:12:28 2011
*nat
:PREROUTING ACCEPT [2313499:277616614]
:POSTROUTING ACCEPT [878281:52700139]
:OUTPUT ACCEPT [878281:52700139]
:xyz - [0:0]
-A xyz –p tcp –j ACCEPT
COMMIT
# Completed on Mon Mar  7 06:12:28 2011

# Generated by iptables-save v1.4.10 on Mon Mar  7 06:12:28 2011
*mangle
:PREROUTING ACCEPT [1533893:158705924]
:INPUT ACCEPT [1527850:158174252]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [404209:35103562]
:POSTROUTING ACCEPT [404209:35103562]
-A INPUT -s 192.168.14.61/32 -p icmp -j ACCEPT
-A FORWARD -j CLASSIFY --set-class 0001:0020
-A OUTPUT -d 192.168.14.61/32 -p icmp -j CLASSIFY --set-class
0001:0020
COMMIT
# Completed on Mon Mar  7 06:12:28 2011

# Generated by iptables-save v1.4.10 on Mon Mar  7 06:12:28 2011
*filter
:INPUT ACCEPT [45811:3619261]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [15294:1046945]
:ABC - [0:0]
-A FORWARD -s 192.168.10.15/32 -p tcp -m recent --update --seconds
100 --name ABC --rsource -j ACCEPT
-A FORWARD -s 192.168.10.0/24 -m recent --update --seconds 100 --
name ABC --rsource -j ACCEPT
COMMIT
```

[3] Iako je Iptables naredbe moguće definirati samo na Linux sustavima, nakon što se dohvate i spreme u datoteku FIRO ih može procesuirati ili na Windows ili na Linux sustavu.

```
# Completed on Mon Mar  7 06:12:28 2011
```

`iptables-save` naredba generira za svaku tablicu (`*raw`, `*nat`, `*mangle`, `*filter`) popis predefiniranih i korisnički definiranih lanaca (indikator '`:`') nakon kojeg slijede Iptables naredbe (indikator '`-A `'), te na kraju oznaka `COMMIT` kao potvrda kraja naredbi.

Budući da je sintaksa fiksna i drugi jednostavniji vatrozidi mogu se optimizirati pomoću Firo programa, naravno uz određene prilagodbe.

6.4. Izlazni podaci - sekvencijalno generiranje rezultata

Firo kao izlaz daje više rezultata. Osnovni rezultat Firo programa je niz datoteka unutar kojih se nalaze optimirana sigurnosna pravila. Na primjer ukoliko je u `config.txt` konfiguracijskoj datoteci definiran kao prefiks „`vatrozid`“, tada će se generirati niz datoteka počevši s `vatrozid_0.txt`, a unutar nje se nalazi početni niz pravila koji je moguće optimizirati. Nakon svakog završenog koraka optimiranja u sklopu kojeg je barem jedno pravilo uklonjeno ili modificirano generira se nova izlazna datoteka `vatrozid_n.txt` i tako sve dok više nije moguće obaviti optimizaciju.

Kako bi se lakše analizirale generirane datoteke Firo generira i jednu datoteku (`vatrozid_log_changes.txt`) u kojoj su popisane sve datoteke s opisom obavljenih optimizacija i prijašnjim te novim sigurnosnim pravilima. U nastavku slijedi primjer jedna takve datoteke:

```
***************    vatrozid_1.txt   *****************
DELETED RULE (no action):
  [Del] -A FORWARD -p icmp -s 2.3.4.5

DELETED RULE (wrong parameters:  -m ttl --ttl-gt 255):
  [Del] -A FORWARD -p all -s 7.8.9.10 -m ttl --ttl-gt 255 -j ACCEPT

CHANGED RULE:
        -A INPUT -d 2.4.8.22 -m multiport --sports 11,12,13,14
           -j DROP
  [New] -A INPUT -d 2.4.8.22 --sport 11:14 -j DROP


***************    vatrozid_2.txt   *****************
RULE SHADOWED AFTER:
        -A FORWARD -d 2.5.8.54 -m iprange --src-range 14.38.15.0/24
           -j DROP
  [DEL] -A FORWARD -d 2.5.8.54 -m iprange --src-range 14.38.15.176-
14.38.15.199 -j LOG


***************    vatrozid_3.txt   *****************
RULE SHADOWED BEFORE:
  [DEL] -A INPUT -p tcp -s 3.4.11.3 -m multiport
```

```
           --sports 2950,2970,2980 -j DROP
        -A INPUT -p tcp -s 3.4.11.3 --sport 2900:3000 -j DROP


***************    vatrozid_4.txt   *****************
DELETED RULE - HAS SAME ACTION AS POLICY (CHAIN):
        -A tcp_packets -p tcp -s 1.1.1.9 -j ACCEPT


***************    vatrozid_5.txt   *****************
MERGE RULES:
    [1] -A INPUT -p tcp -s 31.13.0.0/24 -m ttl --ttl-lt 215 -j DROP
    [2] -A INPUT -p tcp -s 31.13.0.0/24 -m ttl --ttl-eq 215 -j DROP
  [New] -A INPUT -p tcp -s 31.13.0.0/24 -m ttl --ttl-lt 216 -j DROP


***************    vatrozid_6.txt   *****************
REMOVED PARAMETER (src-type) FROM RULE:
[Other] -A INPUT -p tcp -s 4.4.10.111 -m addrtype
          ! --src-type UNICAST,UNREACHABLE,UNSPEC,XRESOLVE -j DROP
        -A INPUT -p tcp -s 4.4.10.111 -m addrtype
          ! --src-type ANYCAST,BLACKHOLE,BROADCAST -j ACCEPT
  [New] -A INPUT -p tcp -s 4.4.10.111 -j ACCEPT


***************    vatrozid_7.txt   *****************
REMOVED REDUNDANT ELEMENTS FROM RULE:
[Other] -A INPUT -p tcp -s 3.3.3.3 -m multiport
           --sports 90,91,92,93,171,172,173,174,175,176 -j DROP
        -A INPUT -p tcp -s 3.3.3.3 -m multiport
           --sports 90,91,92,93,171,172,210,211,212,213,214,215
           -j DROP
  [New] -A INPUT -p tcp -s 3.3.3.3 --sport 210:215 -j DROP


***************    vatrozid_8.txt   *****************
RULE SHADOWED AFTER:
        -A INPUT -p tcp -s 4.4.10.111 -j ACCEPT
  [DEL] -A INPUT -p tcp -s 4.4.10.111 --sport 55:99 -j DROP


***************    vatrozid_9.txt   *****************
DELETED RULE - HAS SAME ACTION AS POLICY (CHAIN):
        -A INPUT -p tcp -s 4.4.10.111 -j ACCEPT


***************    vatrozid_10.txt   *****************

REPOSITION OF LOG RULES


***************     STATISTICS   *****************
Number of rules at start: 18
        *filter, INPUT (10)
        *filter, FORWARD (6)
        *filter, tcp_packets (2)

Number of rules at end: 12
        *filter, INPUT (6)
```

```
        *filter, FORWARD (5)
        *filter, tcp_packets (1)

Number of rules shadowed after: 2
Number of rules shadowed before: 1
Number of last rules removed: 2
Number of merged rules: 2
```

Ukoliko je potrebno analizirati način rada Firo programa, korisnik može odabrati željenu razinu detaljnosti te opis izvođenja programa, procesuiranja ulazne datoteke, uspoređivanja sigurnosnih pravila i njihovog optimiranja pregledati u log datoteci ili izravno na ekranu monitora.

6.5. Glavne programske komponente

Firo program realiziran je u C++ programskom jeziku korištenjem programskih klasa. U sljedećoj tablici dan je osnovni opis implementiranih klasa.

Tablica 13: Programske klase Firo programa

Programska klasa	Opis
cConfiguration	Klasa u kojoj se nalaze svi konfiguracijski podaci: lokacije direktorija, nazivi datoteka, koje lance treba optimizirati, koje optimizacijske metode uključiti, kakve parametre očekivati u sigurnosnim pravilima, popis i klasifikacija akcija i dr.
cConstant	Klasa za potrebu identificiranja konstanti (IP adrese, mreže, portovi i sl.). Trenutačno nema primjernu s ulaznim podacima generiranim `iptables-save` alatom u sklopu kojih se ne nalaze konstante.
cLibrary	Klasa s internim osnovnim funkcijama za potrebe konverzije podataka ili ispisa *error* pogrešaka.
cWeight	Klasa u kojoj se nalaze definirani parametri – sadrži definiciju svakog parametra te se koristi kod prepoznavanja parametara u sigurnosnim pravilima.
cCharField	Klasa koja sadrži znakovni element.
cRule	Klasa u kojoj se pohranjuju osnovni podaci o sigurnosnom pravilu (akcija, parametri uspoređivanja, parametri akcije).
cRuleFrame	Klasa koja obuhvaća sigurnosno pravilo, ali i reference (relacije) tog pravila prema drugim pravilima ispred i nakon njega u lancu.
cPolicy	Klasa koja obuhvaća sva sigurnosna pravila iz jednog lanca.
cParameter	Klasa koja sadrži zajedničke funkcije i procedure za sljedećih 7 klasa navedenih u ovoj tablici.
cString	Klasa koja implementira znakovne parametre i nizove znakovnih vrijednosti te sve akcije koje se provode nad njima (utvrđivanje relacije, utvrđivanje uključenosti, spajanje, uklanjanje redundantnih elemenata).

cIpaddres	Klasa koja implementira parametre u obliku IP adresa, mreža ili IP raspona te sve akcije koje se provode nad njima.
cPortNumber	Klasa koja implementira portove i brojeve, bilo da su u obliku broja, raspona ili niza te sve akcije koje se provode nad njima.
cFlag	Klasa koja implementira TCP zastavice te sve akcije koje se provode nad njima.
cDateTime	Klasa koja implementira datum i vrijeme te sve akcije koje se provode nad njima.
cProtocol	Klasa koja implementira parametar protokola te sve akcije koje se provode nad njim.
cTtl	Klasa koja implementira TTL vrijednosti iz paketa te sve akcije koje se provode nad njima.

Detaljniji opis programskih komponenti nalazi se na kompaktnom disku koji je prilog ovom radu.

6.6. Osnovne operacije nad parametrima

U ovom poglavlju opisane su osnovne operacije koje se obavljaju nad parametrima, a u svrhu optimiranja sigurnosnih pravila. Taj opis je dan budući da je ovaj rad, za razliku od svih drugih analiziranih radova, zasnovan na velikom broju parametara koji mogu poprimiti kompleksne vrijednosti, čime se kompliciraju i akcije.

Kako bi se parametri mogli uspoređivati te se nad njima provoditi određene operacije, oni moraju biti istog tipa (PortNumber, String, IPaddress,...) i istog naziva (*source port*, *ttl*, itd.).

6.6.1. Utvrđivanje preklapanja parametara

Utvrđivanje preklapanja parametara ključno je za određivanje nalaze li se sigurnosna pravila u relaciji (engl. *related*).

6.6.1.1. cString: preklapanja znakovnih parametara

Parametri koji su klasificirani kao znakovni (*string*), dijele se na one koji mogu biti proizvoljne vrijednosti, i na one koji mogu imati ograničeni niz definiranih vrijednosti (tzv. *multivalue*).

Procedura utvrđivanja preklapanja među znakovnim parametrima:

```
ako su parametri multivalue onda
  ako su oba parametra neinvertirana onda
    ako sadrže isti element onda
      preklapaju se;
  inače ako su oba parametra invertirana onda
```

```
    ako postoji element koji ni jedan ne sadrži onda
      preklapaju se;
  inače ako je prvi parametar invertiran onda
    ako drugi parametar ima element koji nema prvi onda
      preklapaju se;
  inače ako je drugi parametar invertiran onda
    ako prvi parametar ima element koji nema drugi onda
      preklapaju se;
inače ako je vrijednost parametara jednaka onda
  ako je inverzija parametra jednaka onda
      preklapaju se;
inače ako je vrijednost parametara različita onda
  ako se radi o sučeljima onda
      procedura za sučelja;
  inače ako je inverzija parametra različita onda
      preklapaju se;
```

U navedenoj proceduri postoji izuzetak za mrežna sučelja gdje se npr. `eth0` i `eth+` preklapaju iako su im vrijednosti različite. Naime, `eth+` označava sva `eth` sučelja pa tako i `eth0`. Stoga je za mrežna sučelja dodan izuzetak kad su vrijednosti različite. U sljedećem slučaju dan je algoritam utvrđivanja preklapanja sučelja kad barem jedno od sučelja ima na kraju oznaku '+':

```
ako prvi parametar (sučelje) ima '+' na kraju onda
  ako je parametrima prefiks isti onda
    ako je prvi neinvertiran ili je drugi invertiran onda
      preklapaju se;
  inače ako parametri nemaju isti prefiks onda
    ako je barem jedan invertiran onda
      preklapaju se;
inače ako drugi parametar-sučelje ima '+' na kraju onda
  ako je parametrima prefiks isti onda
    ako je drugi neinvertiran ili je prvi invertiran onda
      preklapaju se;
  inače ako parametri nemaju isti prefiks onda
    ako je barem jedan invertiran onda
      preklapaju se;
```

6.6.1.2. cProtocol: preklapanje protokola

Posebna klasa za protokol postoji samo iz povijesnih razloga – nastala je prije cString klase spomenute u prethodnom poglavlju. Iako svojim vrijednostima može biti promatrana kao znakovni niz (bez obzira označava li se protokol znakovnim nizom ili brojem).

Procedura utvrđivanja preklapanja protokola:

```
ako su parametri jednaki onda
  ako je inverzija parametra ista onda
```

```
      preklapaju se;
inače ako je jedan od parametara „ALL“ onda
  preklapaju se;
inače ako su parametri različiti onda
  ako je inverzija parametra različita onda
    preklapaju se;
```

U algoritmu je vidljivo da ukoliko jedan od parametara ima vrijednost „ALL“ (hrv. svi), onda se ne provjerava inverzija i parametri se sigurno preklapaju. To je zato što inverzija od „ALL“ nije dozvoljena, čak i kad bi bila, takav parametar ne bi uključivao niti jedan paket jer svaki paket mora biti određenog protokola.

6.6.1.3. cFlag: preklapanje TCP zastavica

Posebna klasa cFlag razvijena je za TCP zastavice (engl. *flags*). To su bitovi u TCP zaglavlju (SYN, ACK, FIN, RST, PSH, URG) koji mogu biti 1 ili 0, a u Iptables pravilima definiraju se zastavice koje se promatraju te moraju li one biti postavljene.

Procedura utvrđivanja preklapanja zastavica:

```
ako su oba parametra bez inverzije onda
  ako ne postoji zastavica koja se provjerava u oba parametra
  i očekivane vrijednosti su joj različite onda
    preklapaju se;
inače ako su oba parametra s inverzijom onda
  ako barem jedan parametar provjerava više od jedne
  zastavice onda
    preklapaju se;
  inače ako parametri provjeravaju različite zastavice onda
    preklapaju se;
  inače ako parametri provjeravaju istu zastavicu pri čemu im
  je očekivana vrijednost ista onda
    preklapaju se;
inače ako je samo prvi parametar s inverzijom onda
  ako postoji zastavica koja se provjerava u prvom
  parametru a ne provjerava se u drugom onda
    preklapaju se;
  inače ako postoji zastavica koja se provjerava u prvom
  parametru a u drugom parametru ima različitu očekivanu
  vrijednost onda
    preklapaju se;
inače ako je samo drugi parametar s inverzijom onda
  ako postoji zastavica koja se provjerava u drugom
  parametru a ne provjerava se u prvom onda
    preklapaju se;
  inače ako postoji zastavica koja se provjerava u drugom
  parametru a u prvom parametru ima različitu očekivanu
  vrijednost onda
```

```
    preklapaju se;
```

6.6.1.4. cPortNumber: preklapanje portova i brojeva

Kod uspoređivanja portova tj. brojeva postoje 3 osnovna tipa vrijednosti parametara: pojedinačna vrijednost, raspon vrijednosti (minimalno 2 vrijednosti) i niz vrijednosti (minimalno 2 vrijednosti). Iako prava implementirana procedura uključuje i usporedbe npr. raspon–pojedinačan, niz-pojedinačan, niz-raspon, radi jednostavnosti u ovom algoritmu ti su primjeri isključeni budući da su načelno jednaki usporedbama pojedinačan-raspon, pojedinačan-niz i raspon-niz.

```
ako je prvi parametar pojedinačna vrijednost onda
  ako je drugi parametar pojedinačna vrijednost onda
    ako su vrijednosti iste onda
      ako je inverzija ista kod oba parametra onda
        preklapaju se;
    inače ako su vrijednosti različite onda
      ako je barem jedan parametar s inverzijom onda
        preklapaju se;
  inače ako je drugi parametar raspon vrijednosti onda
    ako je prvi parametar uključen u raspon onda
      ako je prvi parametar invertiran ili drugi nije onda
        preklapaju se;
    inače ako prvi parametar nije uključen u raspon onda
      ako je prvi ili drugi parametar invertiran onda
        preklapaju se;
  inače ako je drugi parametar niz vrijednosti onda
    ako je prvi parametar uključen u niz onda
      ako je prvi parametar invertiran ili drugi nije onda
        preklapaju se;
    inače ako prvi parametar nije uključen u niz onda
      ako je prvi ili drugi parametar invertiran onda
        preklapaju se;
inače ako je prvi parametar raspon vrijednosti onda
  ako je drugi parametar pojedinačna vrijednost onda
    {recipročna procedura pojedinačan-raspon}
  inače ako je drugi parametar raspon vrijednosti onda
    ako su rasponi identični onda
      ako je inverzija ista kod oba parametra onda
        preklapaju se;
    inače ako je drugi raspon unutar prvog onda
      ako je inverzija ista ili je samo drugi invertiran onda
        preklapaju se;
    inače ako je prvi raspon unutar drugog onda
      ako je inverzija ista ili je samo prvi invertiran onda
        preklapaju se;
    inače ako se rasponi ne preklapaju onda
      ako je barem jedan invertiran onda
        preklapaju se;
    inače ako se rasponi djelomično preklapaju onda
```

```
      preklapaju se;
  ako je drugi parametar niz vrijednost onda
    ako oba parametra nisu invertirana onda
      ako je barem jedan element niza u rasponu onda
        preklapaju se;
    inače ako su oba parametra invertirana onda
      ako sumarno ne sadrže sve parametre onda
        preklapaju se;
    inače ako je samo prvi parametar invertiran onda
      ako barem jedan element niza nije u rasponu onda
        preklapaju se;
    inače ako je samo drugi parametar invertiran onda
      ako barem jedan element raspona nije u nizu onda
        preklapaju se;
inače ako je prvi parametar niz vrijednosti onda
  ako je drugi parametar pojedinačna vrijednost onda
    {recipročna procedura pojedinačan-niz}
  inače ako je drugi parametar raspon vrijednosti onda
    {recipročna procedura raspon-niz}
  inače ako je drugi parametar niz vrijednosti onda
    ako oba parametra nisu invertirana onda
      ako je barem jedan element nizova zajednički onda
        preklapaju se;
    inače ako su oba parametra invertirana onda
      preklapaju se;
    inače ako je samo prvi parametar invertiran onda
      ako barem jedan element drugog nije u prvom nizu onda
        preklapaju se;
    inače ako je samo drugi parametar invertiran onda
      ako barem jedan element prvog nije u drugom nizu onda
        preklapaju se;
```

6.6.1.5. cTtl: preklapanje TTL oznake

Iako se cTTL klasa mogla dijelom uključiti u cPortNumber, ipak tu postoje posebna pravila koja se ne koriste kod cPortNumber klase pa je zadržana posebna klasa. Posebna pravila su „jednako“, „više od“ i „manje od“ usporedbe, od kojih zadnja dva ne smiju biti invertirana.

Procedura utvrđivanja preklapanja TTL parametara ne opisuje slučaj kad je prvi parametar definiran s „manje od“ ili „više od“, a drugi parametar definiran pojedinačnom vrijednosti jer je tu procedura ista kao kad su obrnuti parametri (a taj je slučaj opisan):

```
ako je prvi parametar pojedinačan onda
  ako je drugi parametar pojedinačan onda
    ako su vrijednosti iste onda
      ako je inverzija ista kod oba parametra onda
        preklapaju se;
    inače ako su vrijednosti različite onda
      ako je barem jedan parametar s inverzijom onda
```

```
        preklapaju se;
  inače ako je drugi parametar definiran s „manje od“ ili
  „više od“ onda
    ako je prva vrijednost uklopljena u drugi parametar onda
      ako prvi parametar nije invertiran onda
        preklapaju se;
    inače ako prva vrijednost nije u sklopu drugog parametra
    onda
      Ako je prvi parametar invertiran onda
        preklapaju se;
inače  ako je prvi parametar definiran s „manje od“ ili „više
od“ onda
  ako je drugi parametar pojedinačan onda
    {recipročna procedura pojedinačan - manje/više od}
  inače ako su oba jednako definirana onda
    preklapaju se;
  inače ako je prvi s „manje od“ a drugi s „više od“ onda
    ako prvi ima veću vrijednost od drugog uvećanog za 1 onda
      preklapaju se;
  inače ako je prvi s „više od“ a drugi s „manje od“ onda
    ako drugi ima veću vrijednost od prvog uvećanog za 1 onda
      preklapaju se;
```

6.6.1.6. cIPaddress: preklapanje IP adresa

Utvrđivanje preklapanja među pojedinačnim IP adresama, IP rasponom (mreža je tip raspona) te IP adresom s *wildcard* maskom implementirano je u sklopu cIPaddress klase.

Procedura utvrđivanja preklapanja IP zapisa kad je jedan od zapisa IP adresa:

```
ako je prvi parametar IP adresa onda
  ako je drugi parametar IP adresa onda
    ako su vrijednosti iste onda
      ako je inverzija ista kod oba parametra onda
        preklapaju se;
    inače ako su vrijednosti različite onda
      ako je barem jedan od parametara s inverzijom onda
        preklapaju se;
  inače ako je drugi parametar IP raspon onda
    ako je prvi parametar s inverzijom onda
      preklapaju se;
    inače ako je prvi parametar unutar raspona onda
      ako je drugi parametar bez inverzije onda
        preklapaju se;
    inače ako je prvi parametar izvan raspona onda
      ako je drugi parametar s inverzijom onda
        preklapaju se;
  inače ako je drugi parametar IP adresa s wildcard maskom
  onda
```

```
    ako je prvi parametar s inverzijom onda
      preklapaju se;
    inače ako je prvi parametar u obuhvatu „maske“ onda
      ako je drugi parametar bez inverzije onda
        preklapaju se;
    inače ako je prvi parametar izvan obuhvata „maske“ onda
      ako je drugi parametar s inverzijom onda
        preklapaju se;
```

U nastavku slijedi procedura utvrđivanja preklapanja IP zapisa kad je jedan od zapisa IP raspon. U toj proceduri koristi se i posebna funkcija `range_mask_related()` koja provjerava preklapaju li se neinvertirani IP raspon i IP adresa s *wildcard* maskom, a ukratko je opisana odmah iza sljedeće procedure:

```
ako je prvi parametar IP raspon onda
  ako je drugi parametar IP adresa onda
    {recipročna procedura već opisana}
  inače ako je drugi parametar IP raspon onda
    ako su rasponi identični onda
      ako je inverzija ista kod oba parametra onda
        preklapaju se;
    inače ako je drugi raspon unutar prvog onda
      ako je inverzija ista ili je samo drugi invertiran onda
        preklapaju se;
    inače ako je prvi raspon unutar drugog onda
      ako je inverzija ista ili je samo prvi invertiran onda
        preklapaju se;
    inače ako se rasponi ne preklapaju onda
      ako je barem jedan invertiran onda
        preklapaju se;
    inače ako se rasponi djelomično preklapaju onda
      preklapaju se;
  ako je drugi parametar IP adresa s wildcard maskom onda
    ako su oba parametra bez inverzije onda
      ako je range_mask_related() istinit onda
        preklapaju se;
    inače ako je samo prvi parametar s inverzijom onda
      ako je range_mask_related() istinit za raspon od
      0.0.0.0 do prvog elementa raspona onda
        preklapaju se;
      inače ako je range_mask_related() istinit za raspon od
      elementa iza drugog elementa raspona do 255.255.255.255
      onda
        preklapaju se;
    inače ako su oba parametra s inverzijom onda
      ako je range_mask_related() lažan za raspon od
      0.0.0.0 do elementa ispred prvog elementa raspona onda
        preklapaju se;
      inače ako je range_mask_related() lažan za raspon od
      elementa iza drugog elementa raspona do 255.255.255.255
      onda
        preklapaju se;
```

```
    inače ako je samo drugi parametar s inverzijom onda
      ako je range_mask_related() lažan onda
        preklapaju se;
```

U prethodnom slučaju kad je jedan od parametara IP raspon, a drugi IP adresa s wildcard maskom korištena je posebna procedura `range_mask_related();` u kojoj se osnovnoj IP adresi mijenjaju „*wildcard*“ bitovi na način da se krene od prvog „*wildcard*“ bita koji se mijenja iz 0 u 1. U slučaju da se tako dobivena IP adresa nalazi unutar promatranog raspona, procedura se prekida jer se zapisi preklapaju. Ukoliko se nalazi „ispod“ promatranog raspona, zadržava se postavljena 1 te se kreće sa sljedećim manjim bitom. A ako se nalazi „iznad“ promatranog raspona, postavljeni bit se zamjenjuje s 0 te se kreće sa sljedećim nižim bitom. Time se u maksimalno 32 prolaza (koliko ima bitova u IP adresi) može utvrditi preklapaju li se zapisi.

```
range_mask_related():

ako je (IP_adresa & wildcard_mask) unutar raspona onda
  preklapaju se;
shift := 0x80000000 //binarno:10000000000000000000000000000000

ponavljaj
  // wildcard maski se postavlja „shift“ bit na 1
  nova_wildcard_mask := wildcard_mask | shift;

  ako je nova_wildcard_mask različita od osnovne
  wildcard_mask onda
    // IP adresi se postavlja „shift“ bit na 1
    nova_IP_adresa := IP_adresa | shift;
    ako je nova_IP_adresa unutar raspona onda
      preklapaju se;
    inače ako je nova_IP_adresa veća od raspona onda
      nova_IP_adresa := IP_adresa;
      nova_wildcard_mask := wildcard_mask;
    inače ako je nova_IP_adresa manja od raspona onda
      IP_adresa := nova_IP_adresa;
      wildcard_mask := nova_wildcard_mask;

  shift := shift >> 1; // pomicanje bita udesno
dok je (shift != 0)
```

U nastavku je opisano utvrđivanje preklapanja kad su oba IP zapisa IP adrese s *wildcard* maskom:

```
ako je prvi parametar IP1 adresa s wildcard maskom W1 onda
  ako je drugi parametar IP2 adresa s wildcard maskom W2 onda
    ako su oba parametra s inverzijom[4] onda
```

[4] Najveća IP adresa s *wildcard* maskom može uključivati maksimalno pola adresnog prostora (ako je samo jedan bit fiksan, a svi ostali promjenjivi). Stoga se inverzije dva takva IP zapisa

```
      ako su wildcard maske različite ili imaju više od 1
      fiksnog bita ili su im IP adrese iste onda
        preklapaju se;
    inače ako su oba parametra bez inverzije onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) = (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        preklapaju se;
    inače ako je samo prvi parametar IP1 s inverzijom onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) != (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        preklapaju se;
      inače ako je (inverzna W1 wildcard maska & inverzna W2
      wildcard maska) != (inverzna W1 wildcard maska) onda
        preklapaju se;
    inače ako je samo drugi parametar IP2 s inverzijom onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) != (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        preklapaju se;
      inače ako je (inverzna W1 wildcard maska & inverzna W2
      wildcard maska) != (inverzna W2 wildcard maska) onda
        preklapaju se;
```

Ako se IP raspon može zapisati kao mreža tada se taj IP zapis isto može promatrati i kao *wildcard* maska kojoj su bitovi na početku postavljeni na 1, a bitovi na kraju na 0. Stoga ako je IP raspon ujedno i mreža, tada se za utvrđivanje preklapanja među IP zapisima koristi zadnje opisana usporedba gdje su oba zapisa *wildcard* maske.

6.6.1.7. cDateTime: preklapanje datuma i vremena

cDateTime posebna je klasa namijenjena akcijama povezanim s parametrima datuma i vremena. Iptables datumi i vremena specifični su jer nemaju inverzije, a vremensko razdoblje može biti određeno bez definiranog početka, bez definiranog kraja, ili kao raspon od-do.

Budući da kod Iptables alata za vrijeme i datum nije dozvoljena inverzija, procedura utvrđivanja preklapaju li se parametri datuma i vremena je jednostavna:

```
ako oba parametra nemaju definiran početak onda
  preklapaju se;
inače ako oba parametra nemaju definiran kraj onda
  preklapaju se;
inače ako je prvom parametru početak poslije kraja drugog
parametra onda
```

---

u pravilu uvijek preklapaju. Izuzetak je jedino ako ta dva IP zapisa obuhvaćaju dvije inverzne polovine adresnog prostora.

```
  preklapaju se;
inače ako je prvom parametru kraj prije početka drugog
parametra onda
  preklapaju se;
```

6.6.2. Utvrđivanje međusobne uključenosti parametara

Utvrđivanje međusobne uključenosti parametara ključno je za određivanje je li jedno sigurnosno pravilo u „sjeni ispred“ ili u „sjeni iza“ – a kako bi to bilo ostvareno nužno je da su vrijednosti svih parametara jednog pravila sadržane (engl. *contained*) u vrijednostima pripadnih parametara drugog pravila.

6.6.2.1. cString: međusobna uključenost znakovnih parametara

Parametri koji su klasificirani kao znakovni (*string*), dijele se na one koji mogu biti proizvoljne vrijednosti i na one koji mogu imati ograničeni niz definiranih vrijednosti (tzv. *multivalue*).

Procedura utvrđivanja uključenosti drugog znakovnog parametra u prvi znakovni parametar:

```
ako su parametri multivalue onda
  ako su oba parametra neinvertirana onda
    ako prvi parametar sadrži sve elemente iz drugog
    parametra onda
      drugi parametar je uključen u prvi;
  inače ako su oba parametra invertirana onda
    ako drugi parametar sadrži sve elemente iz prvog
    parametra onda
      drugi parametar je uključen u prvi;
  inače ako je prvi parametar invertiran onda
    ako parametri nemaju ni jedan zajednički element onda
      drugi parametar je uključen u prvi;
  inače ako je drugi parametar invertiran onda
    ako oba parametra sadrže sve moguće elemente onda
      drugi parametar je uključen u prvi;
inače ako je vrijednost parametara jednaka onda
  ako je inverzija parametra jednaka onda
    drugi parametar je uključen u prvi;
inače ako je vrijednost parametara različita onda
  ako se radi o sučeljima onda
    procedura za sučelja;
  inače ako samo prvi parametar ima inverziju onda
    drugi parametar je uključen u prvi;
```

U navedenoj proceduri postoji izuzetak za mrežna sučelja gdje je npr. `eth0` sadržan u `eth+` iako su im vrijednosti različite. Stoga je za mrežna sučelja dodan izuzetak kad su vrijednosti različite. U sljedećem slučaju dan je algoritam utvrđivanja međusobne uključenosti kada barem jedno od sučelja ima oznaku '+':

```
ako prvi parametar (sučelje) ima '+' na kraju onda
  ako je parametrima prefiks isti onda
    ako su parametri bez inverzije onda
      drugi parametar je uključen u prvi;
  inače ako parametri nemaju isti prefiks onda
    ako samo prvi parametar ima inverziju onda
      drugi parametar je uključen u prvi;
inače ako drugi parametar-sučelje ima '+' na kraju onda
  ako je parametrima prefiks isti onda
    ako su oba parametra s inverzijom onda
      drugi parametar je uključen u prvi;
  inače ako parametri nemaju isti prefiks onda
    ako samo prvi parametar ima inverziju onda
      drugi parametar je uključen u prvi;
```

6.6.2.2. cProtocol: međusobna uključenost protokola

Procedura utvrđivanja uključenosti drugog protokola u prvi protokol:

```
ako su parametri jednaki onda
  ako je inverzija parametara ista onda
    drugi parametar je uključen u prvi;
inače ako je prvi parametar „ALL“ onda
  drugi parametar je uključen u prvi;
inače ako su parametri različiti onda
  ako drugi parametar nije „ALL“ onda
    ako samo prvi parametar ima inverziju onda
      drugi parametar je uključen u prvi;
```

Ukoliko prvi parametar ima vrijednost „ALL“ (hrv. svi), onda je drugi parametar obavezno sadržan u prvom.

6.6.2.3. cFlag: međusobna uključenost TCP zastavica

Procedura utvrđivanja je li drugi parametar uključen u prvi parametar:

```
ako su oba parametra bez inverzije onda
  ako ne postoji zastavica koja se provjerava u prvom
  parametru a ne provjerava se u drugom onda
    ako ne postoji zastavica koja se provjerava u prvom
    parametru a u drugom parametru ima različitu očekivanu
    vrijednost onda
      drugi parametar je uključen u prvi;
```

```
inače ako su oba parametra s inverzijom onda
  ako ne postoji zastavica koja se provjerava u drugom
  parametru a ne provjerava se u prvom onda
    ako ne postoji zastavica koja se provjerava u drugom
    parametru a u prvom parametru ima različitu očekivanu
    vrijednost onda
      drugi parametar je uključen u prvi;
inače ako je samo drugi parametar s inverzijom onda
  ako oba parametra imaju samo jednu istu zastavicu pri čemu
  jedan parametar očekuje da bude postavljena, a drugi da ne
  bude (očekivane vrijednosti su različite) onda
    drugi parametar je uključen u prvi;
inače ako je samo prvi parametar s inverzijom onda
  ako postoji zastavica koja se provjerava u oba parametra
  i očekivane vrijednosti su joj različite onda
    drugi parametar je uključen u prvi;
```

6.6.2.4. cPortNumber: međusobna uključenost portova i brojeva

Procedura utvrđivanja uključenosti drugog parametar u prvi parametar:

```
ako je prvi parametar pojedinačna vrijednost onda
  ako je drugi parametar pojedinačna vrijednost onda
    ako su vrijednosti iste onda
      ako je inverzija ista kod oba parametra onda
        drugi parametar je uključen u prvi;
    inače ako su vrijednosti različite onda
      ako je samo prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar raspon vrijednosti onda
    ako je prvi parametar uključen u raspon onda
      ako su oba parametra s inverzijom onda
        drugi parametar je uključen u prvi;
    inače ako prvi parametar nije uključen u raspon onda
      ako je samo prvi parametar invertiran onda
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar niz vrijednosti onda
    ako je prvi parametar uključen u niz onda
      ako su oba parametra s inverzijom onda
        drugi parametar je uključen u prvi;
    inače ako prvi parametar nije uključen u niz onda
      ako je samo prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
inače ako je prvi parametar raspon vrijednosti onda
  ako je drugi parametar pojedinačna vrijednost onda
    ako je drugi parametar uključen u raspon onda
      ako su oba parametra bez inverzije onda
        drugi parametar je uključen u prvi;
    inače ako drugi parametar nije uključen u raspon onda
      ako je samo prvi parametar invertiran onda
        drugi parametar je uključen u prvi;
```

```
  inače ako je drugi parametar raspon vrijednosti onda
    ako su rasponi identični onda
      ako je inverzija ista kod oba parametra onda
        drugi parametar je uključen u prvi;
    inače ako je drugi raspon unutar prvog onda
      ako su oba parametra bez inverzije onda
        drugi parametar je uključen u prvi;
    inače ako je prvi raspon unutar drugog onda
      ako su oba parametra s inverzijom onda
        drugi parametar je uključen u prvi;
    inače ako se rasponi ne preklapaju onda
      ako je samo prvi parametar invertiran onda
        drugi parametar je uključen u prvi;
    inače ako se rasponi djelomično preklapaju onda
      ako je samo drugi parametar invertiran onda
        ako rasponi zajedno pokrivaju sve moguće vrijednosti
        onda
          drugi parametar je uključen u prvi;
  inače ako je drugi parametar niz vrijednosti onda
    ako oba parametra nisu invertirana onda
      ako su svi elementi niza u rasponu onda
        drugi parametar je uključen u prvi;
    inače ako su oba parametra invertirana onda
      ako je cijeli raspon u nizu onda
        drugi parametar je uključen u prvi;
    inače ako je samo prvi parametar invertiran onda
      ako svi elementi niza nisu u rasponu onda
        drugi parametar je uključen u prvi;
    inače ako je samo drugi parametar invertiran onda
      ako raspon i niz zajedno pokrivaju sve moguće
      vrijednosti onda
        drugi parametar je uključen u prvi;
inače ako je prvi parametar niz vrijednosti onda
  ako je drugi parametar pojedinačna vrijednost onda
    ako je drugi parametar uključen u niz onda
      ako su oba parametra bez inverzije onda
        drugi parametar je uključen u prvi;
    inače ako prvi parametar nije uključen u niz onda
      ako je samo prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar raspon vrijednosti onda
    ako oba parametra nisu invertirana onda
      ako su svi elementi raspona u nizu onda
        drugi parametar je uključen u prvi;
    inače ako su oba parametra invertirana onda
      ako su svi elementi niza u rasponu onda
        drugi parametar je uključen u prvi;
    inače ako je samo prvi parametar invertiran onda
      ako svi elementi niza nisu u rasponu onda
        drugi parametar je uključen u prvi;
    inače ako je samo drugi parametar invertiran onda
      ako raspon i niz zajedno pokrivaju sve moguće
      vrijednosti onda
```

```
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar niz vrijednosti onda
    ako oba parametra nisu invertirana onda
      ako su svi elementi drugog niza u prvom nizu onda
        drugi parametar je uključen u prvi;
    inače ako su oba parametra invertirana onda
      ako su svi elementi prvog niza u drugom nizu onda
        drugi parametar je uključen u prvi;
    inače ako je samo prvi parametar invertiran onda
      ako nizovi nemaju ni jedan zajednički element onda
        drugi parametar je uključen u prvi;
```

6.6.2.5. cTtl: međusobna uključenost TTL oznake

Procedura utvrđivanja međusobne uključenosti TTL parametara (kod Iptables alata inverzija je dozvoljena samo uz pojedinačno definiranu TTL vrijednost):

```
ako je prvi parametar pojedinačan onda
  ako je drugi parametar pojedinačan onda
    ako su vrijednosti iste onda
      ako je inverzija ista kod oba parametra onda
        drugi parametar je uključen u prvi;
    inače ako su vrijednosti različite onda
      ako je samo prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar definiran s „manje od“ onda
    ako je vrijednost prvog parametra manja od vrijednosti
    drugog parametra onda
      ako je prvi parametar bez inverzije onda
        ako je vrijednost prvog parametra 0, a drugog 1 onda
          drugi parametar je uključen u prvi;
    inače ako prva vrijednost nije u sklopu drugog parametra
    onda
      ako je prvi parametar invertiran onda
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar definiran s „više od“ onda
    ako je vrijednost prvog parametra veća od vrijednosti
    drugog parametra onda
      ako je prvi parametar bez inverzije onda
        ako je vrijednost prvog parametra maksimalna, a
        drugog za 1 manja onda
          drugi parametar je uključen u prvi;
    inače ako prva vrijednost nije u sklopu drugog
    parametra onda
      ako je prvi parametar invertiran onda
        drugi parametar je uključen u prvi;
inače ako je prvi parametar definiran s „manje od“ onda
  ako je drugi parametar pojedinačan onda
    ako je vrijednost drugog parametra manja od vrijednosti
```

```
    prvog parametra onda
      ako je drugi parametar bez inverzije onda
        drugi parametar je uključen u prvi;
      inače ako je drugi parametar s inverzijom onda
        ako je vrijednost oba parametra maksimalna onda
          drugi parametar je uključen u prvi
  inače ako je drugi parametar definiran s „manje od“ onda
    ako je vrijednost drugog parametra manja od vrijednosti
    prvog parametra onda
      drugi parametar je uključen u prvi;
inače ako je prvi parametar definiran s „više od“ onda
  ako je drugi parametar pojedinačan onda
    ako je vrijednost drugog parametra veća od vrijednosti
    prvog parametra onda
      ako je drugi parametar bez inverzije onda
        drugi parametar je uključen u prvi;
      inače ako je drugi parametar s inverzijom onda
        ako je vrijednost oba parametra minimalna (0) onda
          drugi parametar je uključen u prvi;
  inače ako je drugi parametar definiran s „više“ onda
    ako je vrijednost drugog parametra veća od vrijednosti
    prvog parametra onda
      drugi parametar je uključen u prvi;
```

6.6.2.6. cIPaddress: međusobna uključenost IP adresa

U sklopu cIPaddress klase obavlja se utvrđivanje međusobnih uključenosti pojedinačne IP adrese, IP raspon (mreža je tip raspona) te IP adrese s *wildcard* maskom.

Procedura utvrđivanja uključenosti IP zapisa u pojedinačnoj IP adresi:

```
ako je prvi parametar IP adresa onda
  ako je drugi parametar IP adresa onda
    ako su vrijednosti iste onda
      ako je inverzija ista kod oba parametra onda
        drugi parametar je uključen u prvi;
    inače ako su vrijednosti različite onda
      ako je prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar IP raspon onda
    ako je prvi parametar unutar raspona onda
      ako su oba parametra s inverzijom onda
        drugi parametar je uključen u prvi;
    inače ako je prvi parametar izvan raspona onda
      ako je prvi parametar s inverzijom, a drugi bez onda
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar IP adresa s wildcard maskom
  onda
    ako je prvi parametar u obuhvatu „maske“ onda
      ako su oba parametra s inverzijom onda
```

```
        drugi parametar je uključen u prvi;
    inače ako je prvi parametar izvan obuhvata „maske“ onda
      ako je samo prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
```

U nastavku slijedi procedura utvrđivanja međusobne uključenosti IP zapisa kad je prvi od zapisa IP raspon. U toj proceduri koristi se i funkcija `range_mask_related()` prethodno opisana u poglavlju 6.6.1.6, a koja provjerava preklapaju li se neinvertirani IP raspon i IP adresa s *wildcard* maskom. Također, koriste se i funkcije `range_in_mask()` te `mask_in_range()` koje provjeravaju nalazi li se IP raspon u IP zapisu s *wildcard* maskom i obratno.

```
ako je prvi parametar IP raspon onda
  ako je drugi parametar IP adresa onda
    ako je drugi parametar unutar raspona onda
      ako su oba parametra bez inverzije onda
        drugi parametar je uključen u prvi;
    inače ako je drugi parametar izvan raspona onda
      ako je samo prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
  ako je drugi parametar IP raspon onda
    ako su rasponi identični onda
      ako je inverzija ista kod oba parametra onda
        drugi parametar je uključen u prvi;
    inače ako je drugi raspon unutar prvog onda
      ako su oba parametra bez inverzije onda
        drugi parametar je uključen u prvi;
    inače ako je prvi raspon unutar drugog onda
      ako su oba parametra s inverzijom onda
        drugi parametar je uključen u prvi;
    inače ako se rasponi ne preklapaju onda
      ako je samo prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
  ako je drugi parametar IP adresa s wildcard maskom onda
    ako su oba parametra bez inverzije onda
      ako je mask_in_range() istinit onda
        drugi parametar je uključen u prvi;
    inače ako su oba parametra s inverzijom onda
      ako je range_in_mask() istinit onda
        drugi parametar je uključen u prvi;
    inače ako je samo prvi parametar s inverzijom onda
      ako je mask_range_related() lažan onda
        drugi parametar je uključen u prvi;
    inače ako je samo drugi parametar s inverzijom onda
      ako raspon čini više od pola adresnog prostora i
      uključuje ili minimalnu ili maksimalnu IP adresu onda
        ako raspon uključuje 0.0.0.0 onda
          ako je range_in_mask() istinit za raspon od 0.0.0.0
          do elementa ispred prvog elementa originalnog
          raspona onda
            drugi parametar je uključen u prvi;
```

```
        inače ako raspon uključuje 255.255.255.255 onda
          ako je range_in_mask() istinit za raspon od
          elementa iza drugog elementa originalnog raspona do
          255.255.255.255 onda
            drugi parametar je uključen u prvi;
```

U prethodnom slučaju, za utvrđivanje uključenosti IP adrese s *wildcard* maskom u IP raspon korištena je procedura `mask_in_range()`. Kako bi se utvrdilo je li IP zapis adrese s *wildcard* maskom uključen u IP raspon potrebno je izračunati najmanju i najveću moguću vrijednost koja se može dobiti uz danu adresu i *wildcard* masku. Ako se one nalaze u danom rasponu tada se i sve druge IP adrese iz danog zapisa nalaze uključene u rasponu:

```
mask_in_range():

// izračun najmanje IP adrese iz IP zapisa s wildcard maskom
min_ip = IP_adresa & wildcard_mask;

// izračun najveće IP adrese iz IP zapisa s wildcard maskom
max_ip = IP_adresa | (~ wildcard_mask);

ako se i min_ip i max_ip nalaze u rasponu onda
  IP zapis s wildcard maskom je uključen u IP raspon;
```

Također, korištena je i procedura `range_in_mask()` za utvrđivanje je li IP raspon uključen u IP zapis adrese s *wildcard* maskom. U tu svrhu i za IP raspon izračunava se osnovna IP „mrežna“ adresa te „mrežna“ maska koje obuhvaćaju cijeli raspon. Kako bi se ustanovila uključenost potrebno je zadovoljiti sljedeća dva uvjeta. Prvi uvjet je da „mrežna“ IP adresa raspona (kojoj su logičkom AND operacijom ostavljeni samo zajednički fiksni bitovi iz *wildcard* maske i „mrežne“ maske) bude jednaka „*wildcard*“ IP adresi (kojoj su logičkom AND operacijom isto ostavljeni samo zajednički fiksni bitovi iz *wildcard* maske i „mrežne“ maske). Drugi uvjet za međusobnu uključenost je da su svi proizvoljni bitovi iz „mrežne“ maske ujedno i proizvoljni bitovi u *wildcard* maski. Algoritmom to se utvrđuje na sljedeći način:

```
range_in_mask():

// izračun mrežne maske iz IP raspona (IP1-IP2) XOR akcijom
mrezna_maska := IP1 xor IP2;
korak := 32;

ponavljaj
  korak := korak - 1;
dok ( (mrezna_maska >> korak) == 1)

mrezna_maska := mrezna_maska & (4294967295 << korak) ;

mrezni_IP := IP1 & mrezna_maska;

ako je ((mrezni_IP & wildcard maska & mrezna_maska) ==
```

```
(wildcard_IP & wildcard maska & mrezna_maska) ) onda
  ako je ((~wildcard maska & ~mrezna_maska) == ~mrezna_maska)
  onda
    IP raspon je uključen u IP zapis s wildcard maskom;
```

U nastavku je opisano utvrđivanje međusobne uključenosti kad je prvi od zapisa IP adresa s *wildcard* maskom:

```
ako je prvi parametar IP1 adresa s wildcard maskom W1 onda
  ako je drugi parametar pojedinačna IP adresa onda
    ako je drugi parametar u obuhvatu „maske“ onda
      ako su oba parametra bez inverzije onda
        drugi parametar je uključen u prvi;
    inače ako je drugi parametar izvan obuhvata „maske“ onda
      ako je samo prvi parametar s inverzijom onda
        drugi parametar je uključen u prvi;
  inače ako je drugi parametar IP raspon onda
    ako su oba parametra bez inverzije onda
      ako je range_in_mask() istinit onda
        drugi parametar je uključen u prvi;
    inače ako su oba parametra s inverzijom onda
      ako je mask_in_range() istinit onda
        drugi parametar je uključen u prvi;
    inače ako je samo prvi parametar s inverzijom onda
      ako je mask_range_related() lažan onda
        drugi parametar je uključen u prvi;
    inače ako je samo drugi parametar s inverzijom onda
      ako raspon čini više od pola adresnog prostora i
      uključuje ili minimalnu ili maksimalnu IP adresu onda
        ako raspon uključuje 0.0.0.0 onda
          ako je range_in_mask() istinit za raspon od 0.0.0.0
          do elementa ispred prvog elementa originalnog
          raspona onda
            drugi parametar je uključen u prvi;
        inače ako raspon uključuje 255.255.255.255 onda
          ako je range_in_mask() istinit za raspon od
          elementa iza drugog elementa originalnog raspona do
          255.255.255.255 onda
            drugi parametar je uključen u prvi;
  inače ako je drugi parametar IP2 adresa s wildcard maskom
  W2 onda
    ako su oba parametra s inverzijom onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) = (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        ako je (~W1 wildcard maska & ~W2 wildcard maska) =
        ~W1 wildcard maska onda
          drugi parametar je uključen u prvi;
    ako su oba parametra bez inverzije onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) = (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        ako je (W1 wildcard maska & W2 wildcard maska) =
```

```
        W1 wildcard maska onda
          drugi parametar je uključen u prvi;
    ako je samo prvi parametar IP1 s inverzijom onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) != (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        drugi parametar je uključen u prvi;
    ako je samo drugi parametar IP2 s inverzijom onda
      ako su wildcard maske iste i imaju samo 1 fiksni bit
      i imaju istu IP adresu onda
        drugi parametar je uključen u prvi;
```

Ako se IP raspon može zapisati kao mreža tada se taj IP zapis isto može promatrati kao *wildcard* maska kojoj su bitovi na početku postavljeni na 1, a bitovi na kraju na 0. Stoga ako je jedan parametar IP raspon koji je ujedno i mreža, a drugi IP adresa s *wildcard* maskom, tada se za utvrđivanje međusobne uključenosti među IP zapisima koristi zadnje opisana usporedba gdje su oba zapisa IP adrese s *wildcard* maskama.

6.6.2.7. cDateTime: međusobna uključenost datuma i vremena

Budući da kod Iptables alata za vrijeme i datum nije dozvoljena inverzija, procedura utvrđivanja međusobne uključenosti parametara datuma i vremena ne obrađuje slučajeve kad je moguća inverzija:

```
ako prvi parametar nema definiran početak onda
  ako je kraj drugog definiran i prije je ili je jednak kraju
  prvog parametra onda
    drugi parametar je uključen u prvi;
inače ako prvi parametar nema definiran kraj onda
  ako je početak drugog definiran i poslije je ili je jednak
  početku prvog parametra onda
    drugi parametar je uključen u prvi;
inače ako oba parametra imaju definiran početak i kraj onda
  ako je početak drugog poslije ili jednak početku prvog
  parametra onda
    ako je kraj drugog prije ili jednak kraju prvog parametra
    onda
      drugi parametar je uključen u prvi;
```

6.6.3. Spajanje parametara

Jedan dio Firo optimizatora je spajanje sigurnosnih pravila. U tu svrhu nužno je da su svi parametri identični osim jednog po kojem se obavlja spajanje. Ukoliko se prvom pravilu modificirao (ili uklonio) promatrani parametar, drugo pravilo se može ukloniti iz liste.

Na početku svakog algoritma spajanja provjerava se je li jedan parametar možda uključen u drugi, ako jest, spajanje se obavlja odmah u tom koraku:

```
ako je drugi parametar uključen u prvi onda
  prvi parametar ostaje isti;
  dozvoljeno uklanjanje drugog pravila;
inače ako je prvi parametar uključen u drugi onda
  prvi parametar postaje drugi parametar;
  dozvoljeno uklanjanje drugog pravila;
```

U nastavku su opisani algoritmi spajanja različitih tipova parametara koji nisu međusobno uključeni jedan u drugi.

6.6.3.1. cString: spajanje znakovnih parametara

Procedura spajanja znakovnih parametara (mogu biti proizvoljne vrijednosti ili imati ograničen niz definiranih vrijednosti - tzv. *multivalue* znakovni parametri):

```
ako su parametri multivalue onda
  ako su oba parametra neinvertirana onda
    ako zajedno pokrivaju sve moguće elemente (vrijednosti)
    onda
      prvi parametar se uklanja;
      dozvoljeno uklanjanje drugog pravila;
    inače
      parametrima se spajaju elementi (bez duplikata);
      dozvoljeno uklanjanje drugog pravila;
    ako dobiveni parametri čine više od polovice svih mogućih
    elemenata onda
      prvi parametar se negira, te se u njega upisuju svi
      ostali elementi (nesadržani u dobivenom parametru);
      dozvoljeno uklanjanje drugog pravila;
  inače ako su oba parametra invertirana onda
    ako parametri nemaju zajedničku vrijednost onda
      prvi parametar se uklanja;
      dozvoljeno uklanjanje drugog pravila;
    inače
      iz prvog parametra se uklanjaju svi elementi koji
      se ne nalaze u drugom parametru;
      dozvoljeno uklanjanje drugog pravila;
  inače ako je prvi parametar invertiran onda
    ako drugi parametar ima sve elemente koji su dio prvog
    parametra onda
      prvi parametar se uklanja;
      dozvoljeno uklanjanje drugog pravila;
    inače ako parametri imaju barem jedan zajednički element
    onda
      iz prvog parametra se uklanjaju svi elementi koji
      se nalaze u drugom parametru;
      dozvoljeno uklanjanje drugog pravila;
```

```
  inače ako je drugi parametar invertiran onda
    ako drugi parametar ima sve elemente koji su dio prvog
    parametra onda
      prvi parametar se uklanja;
      dozvoljeno uklanjanje drugog pravila;
    inače
      prvi parametar se negira, te se u njega upisuju svi
      elementi iz drugog parametra koji nisu dio prvog;
      dozvoljeno uklanjanje drugog pravila;
inače ako je vrijednost parametara jednaka onda
  ako je inverzija parametara različita onda
    prvi parametar se uklanja;
    dozvoljeno uklanjanje drugog pravila;
inače ako je vrijednost parametara različita onda
  ako su oba parametra s inverzijom onda
    prvi parametar se uklanja;
    dozvoljeno uklanjanje drugog pravila;
```

U navedenoj proceduri postoji izuzetak za limitirajuće parametre kod kojih se spajanje obavlja na način da kad je ista vremenska jedinica – ograničenja se zbrajaju. Npr. limit `30/minute` i `20/minute` daju `50/minute`.

6.6.3.2. cProtocol: spajanje protokola

Procedura spajanja protokola:

```
ako su parametri jednaki onda
  ako je inverzija parametara različita onda
    ako je prvi parametar bez inverzije onda
      prvi parametar poprima ALL vrijednost;
      dozvoljeno uklanjanje drugog pravila;
    inače ako je prvi parametar s inverzijom onda
      prvi parametar poprima ALL vrijednost (bez inverzije);
      dozvoljeno uklanjanje drugog pravila;
```

6.6.3.3. cFlag: spajanje TCP zastavica

Spajanje parametara TCP zastavica:

```
ako su oba parametra bez inverzije onda
  ako parametri provjeravaju samo jednu zastavicu te svaki
  parametar očekuje različitu vrijednost onda
    prvi parametar se uklanja;
    dozvoljeno uklanjanje drugog pravila;
  inače ako parametri provjeravaju više istih zastavica te za
  sve očekuju istu vrijednost, osim za jednu za koju svaki
  parametar očekuje različitu vrijednost onda
    iz prvog parametra se uklanja provjeravana zastavica koja
    kod drugog parametra ima različitu očekivanu vrijednost;
```

```
      dozvoljeno uklanjanje drugog pravila;
inače ako su oba parametra s inverzijom onda
  ako postoji zastavica koja se provjerava u oba parametra
  i očekivane vrijednosti su joj različite onda
    prvi parametar se uklanja;
    dozvoljeno uklanjanje drugog pravila;
inače ako je samo drugi parametar s inverzijom onda
  ako ne postoji zastavica koja se provjerava u prvom
  parametru a ne provjerava se u drugom onda
    ako ne postoji zastavica koja se provjerava u prvom
    Parametru a u drugom parametru ima različitu očekivanu
    vrijednost onda
      prvi parametar se uklanja;
      dozvoljeno uklanjanje drugog pravila;
inače ako je samo prvi parametar s inverzijom onda
  ako ne postoji zastavica koja se provjerava u drugom
  Parametru a ne provjerava se u prvom onda
    ako ne postoji zastavica koja se provjerava u drugom
    parametru a u prvom parametru ima različitu očekivanu
    vrijednost onda
      prvi parametar se uklanja;
      dozvoljeno uklanjanje drugog pravila;
```

6.6.3.4. cPortNumber: spajanje portova i brojeva

Kod spajanja portova tj. brojeva postoje 3 osnovna tipa vrijednosti parametara: pojedinačna vrijednost, raspon vrijednosti (minimalno 2 vrijednosti) i niz vrijednosti (minimalno 2 vrijednosti). U nastavku je dan algoritam za spajanje pojedinačnog parametra s pojedinačnim parametrom, rasponom i nizom:

```
ako je prvi parametar pojedinačna vrijednost onda
  ako je drugi parametar pojedinačna vrijednost onda
    ako su vrijednosti iste onda
      ako je inverzija različita onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako su vrijednosti različite onda
      ako su oba parametra bez inverzije onda
        ako je prvi parametar za 1 manji ili veći od drugog
        onda
          prvi parametar spaja se s drugim u raspon;
          dozvoljeno uklanjanje drugog pravila;
        inače ako je dozvoljena konverzija u niz  onda
          prvi parametar postaje niz i dodaje mu se drugi;
          dozvoljeno uklanjanje drugog pravila;
      inače ako su oba parametra s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
  inače ako je drugi parametar raspon vrijednosti onda
    ako je prvi parametar uključen u raspon onda
```

```
      ako je samo drugi parametar s inverzijom onda
        ako je prvi parametar ujedno rubni element raspona
        onda
          ako drugi parametar čine samo 2 elementa onda
            prvi parametar postaje drugi rubni element drugog
            parametra s inverzijom;
            dozvoljeno uklanjanje drugog pravila;
          inače ako drugi parametar čini više od 2 elementa
          onda
            prvi parametar postaje drugi parametar (raspon)
            iz kojeg se uklanja rubni element koji ima
            vrijednost kao i prvi parametar;
            dozvoljeno uklanjanje drugog pravila;
      inače ako je samo prvi parametar s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako prvi parametar nije uključen u raspon onda
      ako su oba parametra bez inverzije onda
        ako je prvi parametar jednak minimalnoj dozvoljenoj
        vrijednosti, a raspon čini sve preostale vrijednosti
        (do maksimalno dozvoljene vrijednosti) onda
          prvi parametar se uklanja;
          dozvoljeno uklanjanje drugog pravila;
        inače ako je prvi parametar jednak maksimalnoj
        dozvoljenoj vrijednosti, a raspon čini sve preostale
        vrijednosti (od minimalne dozvoljene vrijednosti)
        onda
          prvi parametar se uklanja;
          dozvoljeno uklanjanje drugog pravila;
        inače ako je prvi parametar za 1 manji od manjeg
        rubnog elementa raspona, ili za 1 veći od većeg
        rubnog elementa raspona onda
          prvi parametar postaje drugi parametar (raspon)
          proširen prvim parametrom;
          dozvoljeno uklanjanje drugog pravila;
        inače ako je dozvoljena konverzija u niz te je
        veličina raspona za najmanje 1 manja od optimalne
        definirane veličine niza onda
          prvi parametar postaje niz i dodaje mu se svi
          elementi iz drugog parametra;
          dozvoljeno uklanjanje drugog pravila;
      inače ako su oba parametra s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
  inače ako je drugi parametar niz vrijednosti onda
    ako su oba parametra bez inverzije onda
      ako je prvi parametar uključen u niz onda
        prvi parametar postaje drugi parametar;
        dozvoljeno uklanjanje drugog pravila;
      inače ako prvi parametar nije uključen u niz onda
        ako niz zajedno s prvim parametrom čini raspon onda
          prvi parametar postaje raspon kojem je prvi
          parametar najmanja vrijednost iz oba parametra;
```

```
          dozvoljeno uklanjanje drugog pravila;
        inače ako je dozvoljeno generiranje niza te broj
        elemenata u drugom parametru nije veći od maksimalnog
        onda
          prvi parametar i elementi iz drugog parametra se
          spajaju u novi niz;
          dozvoljeno uklanjanje drugog pravila;
    inače ako je samo drugi parametar invertiran onda
      ako je prvi parametar uključen u niz onda
        ako se drugi parametar sastoji od dva elementa onda
          prvi parametar poprima vrijednost drugog elementa
          iz niza, te se primjenjuje inverzija;
          dozvoljeno uklanjanje drugog pravila;
        inače
          prvi parametar postaje drugi parametar iz kojeg se
          uklanja prvi parametar;
          dozvoljeno uklanjanje drugog pravila;
    inače ako je samo prvi parametar invertiran onda
      ako je prvi parametar uključen u niz onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako su oba parametra s inverzijom onda
      ako prvi parametar nije uključen u niz onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
```

Algoritam za spajanje parametra koji je raspon vrijednosti s parametrima raspona i niza vrijednosti (nije naveden algoritam za spajanje raspona i pojedinačnog parametra jer je to opisano u prethodnom algoritmu):

```
ako je prvi parametar raspon vrijednosti onda
  ako je drugi parametar pojedinačna vrijednost onda
    {recipročna procedura pojedinačan-raspon}
  inače ako je drugi parametar raspon vrijednosti onda
    ako su rasponi identični onda
      ako je inverzija različita kod parametara onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je drugi raspon unutar prvog onda
      ako je samo drugi parametar s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je samo prvi parametar s inverzijom onda
        ako je parametrima jedan rubni element isti onda
          prvi parametar postaje smanjeni raspon koji ne
          sadrži drugi raspon;
          dozvoljeno uklanjanje drugog pravila;
    inače ako je prvi raspon unutar drugog onda
      ako je samo prvi parametar s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je samo drugi parametar s inverzijom onda
        ako je parametrima jedan rubni element isti onda
```

```
          prvi parametar postaje drugi parametar (raspon)
          smanjen za raspon originalnog prvog parametra;
          dozvoljeno uklanjanje drugog pravila;
    inače ako se rasponi ne preklapaju onda
      ako su oba parametra bez inverzije onda
        ako su parametri susjedni rasponi onda
          prvi parametar se spaja s drugim parametrom;
          dozvoljeno uklanjanje drugog pravila;
        inače ako prvi parametar uključuje minimalnu, a drugi
        maksimalnu vrijednost, ili obrnuto onda
          prvi parametar postaje inverzni međuraspon;
          dozvoljeno uklanjanje drugog pravila;
        inače ako je dozvoljeno generiranje niza te broj
        elemenata u parametrima nije veći od optimalnog
        onda
          prvi parametar postaje niz sastavljen od svih
          elemenata dvaju originalnih parametara;
          dozvoljeno uklanjanje drugog pravila;
      inače ako su oba parametra s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako se rasponi djelomično preklapaju onda
      ako su oba parametra bez inverzije onda
        prvi parametar se spaja s drugim parametrom u novi
        raspon;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je samo drugi parametar s inverzijom onda
        prvi parametar postaje invertirani drugi parametar iz
        kojeg je uklonjen preklapajući međuraspon;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je samo prvi parametar s inverzijom onda
        iz prvog parametra se uklanja preklapajući
        međuraspon;
        dozvoljeno uklanjanje drugog pravila;
      inače ako su oba parametra s inverzijom onda
        prvi parametar postaje preklapajući međuraspon dvaju
        parametara;
        dozvoljeno uklanjanje drugog pravila;
  inače ako je drugi parametar niz vrijednosti onda
    ako su oba parametra bez inverzije onda
      ako se spajanjem prvog i drugog parametra može dobiti
      neprekinuti raspon onda
        prvi parametar postaje raspon koji uključuje i
        elemente drugog parametra;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je dozvoljeno generiranje niza te broj
      elemenata u parametrima nije veći od maksimalnog onda
        prvi parametar postaje niz sastavljen od svih
        elemenata dvaju originalnih parametara;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je samo prvi parametar s inverzijom onda
      ako su svi elementi raspona sadržani u drugom parametru
      onda
```

```
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako su oba parametra s inverzijom onda
      ako prvi i drugi parametar nemaju ni jedan
      zajednički element onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače
        prvi parametar postaje niz elemenata sadržanih u oba
        parametra (ako je moguće generira se raspon);
        dozvoljeno uklanjanje drugog pravila;
    inače ako je samo drugi parametar s inverzijom onda
      ako su svi elementi niza sadržani u rasponu onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače
        prvi parametar postaje drugi parametar iz kojeg se
        uklanjaju svi elementi iz početnog prvog parametra
        (ako je moguće generira se raspon);
        dozvoljeno uklanjanje drugog pravila;
```

Algoritam za spajanje parametra koji je niz vrijednosti s parametrom niza vrijednosti (nije naveden algoritam za spajanje niza s pojedinačnim parametrom i rasponom jer je to opisano u prethodna dva algoritma):

```
ako je prvi parametar niz vrijednosti onda
  ako je drugi parametar pojedinačna vrijednost onda
    {recipročna procedura pojedinačan-niz}
  inače ako je drugi parametar raspon vrijednost onda
    {recipročna procedura raspon-niz}
  inače ako je drugi parametar niz vrijednosti onda
    ako oba parametra nisu invertirana onda
      ako se spajanjem nizova može dobiti raspon onda
        prvi parametar postaje raspon elemenata sadržanih u
        oba parametra;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je sumarni broj elemenata u nizovima manji od
      maksimalnog dozvoljenog onda
        prvom parametru dodaju se elementi iz drugog niza;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je samo prvi parametar invertiran onda
      ako parametri nemaju zajednički element onda
        prvi parametar se ne mijenja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako su svi elementi prvog parametra sadržani i u
      drugom parametru onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače
        iz prvog parametra uklanjaju se svi elementi sadržani
        u drugom parametru (ako je moguće generira se
        raspon);
        dozvoljeno uklanjanje drugog pravila;
```

```
    inače ako je samo drugi parametar invertiran onda
      ako su svi elementi drugog parametra sadržani i u
      prvom parametru onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače
        prvi parametar postaje drugi parametar iz kojeg se
        uklanjaju svi elementi sadržani u originalnom prvom
        parametru (ako je moguće generira se raspon);
        dozvoljeno uklanjanje drugog pravila;
    inače ako su oba parametra invertirana onda
      ako parametri nemaju zajednički element onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače
        prvi parametar postaje niz elemenata sadržanih u oba
        parametra (ako je moguće generira se raspon);
        dozvoljeno uklanjanje drugog pravila;
```

6.6.3.5. cTtl: spajanje TTL parametara

Procedura spajanja TTL parametara usporedbe (budući da Iptables ne dozvoljavaju kombiniranje inverzije s „manje od“ ili „više od“ ni u sljedećem algoritmu ti slučajevi nisu pokriveni):

```
ako je prvi parametar pojedinačan onda
  ako je drugi parametar pojedinačan onda
    ako su vrijednosti iste onda
      ako je inverzija kod parametara različita onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako su vrijednosti različite onda
      ako su oba parametra s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako parametri imaju vrijednosti 0 i 1 onda
        prvi parametar postaje „manje od“ 2;
        dozvoljeno uklanjanje drugog pravila;
      inače ako parametri imaju vrijednosti 255 i 254 onda
        prvi parametar postaje „manje od“ 2;
        dozvoljeno uklanjanje drugog pravila;
  inače ako je drugi parametar definiran s „manje od“ onda
    ako je prvi parametar bez inverzije onda
      ako je prvi parametar jednak max. vrijednosti
      255, a drugi „manji od“ 255 onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako imaju istu vrijednost onda
        prvi parametar postaje drugi parametar „manji od“
        pri čemu je stara vrijednost uvećana za 1;
        dozvoljeno uklanjanje drugog pravila;
```

```
    inače ako je prvi parametar s inverzijom onda
      ako je vrijednost prvog parametra manja od vrijednosti
      drugog parametra onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
  inače ako je drugi parametar definiran s „više od“ onda
    ako je prvi parametar bez inverzije onda
      ako je prvi parametar jednak min. vrijednosti
      0, a drugi „više od“ 0 onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako imaju istu vrijednost onda
        prvi parametar postaje drugi parametar „veći od“
        pri čemu je stara vrijednost umanjena za 1;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je prvi parametar s inverzijom onda
      ako je vrijednost prvog parametra veća od vrijednosti
      drugog parametra onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
inače  ako je prvi definiran s „manje od“ onda
  ako je drugi parametar pojedinačan onda
    {recipročna procedura spajanja pojedinačan - manje od}
  inače ako je drugi definiran s „više od“ onda
    ako im je vrijednost jednaka onda
      prvi parametar postaje invertirana pojedinačna
      vrijednost;
      dozvoljeno uklanjanje drugog pravila;
    inače ako je vrijednost prvog parametra veća od
    vrijednosti drugog parametra onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
inače  ako je drugi definiran s „više od“ onda
  ako je drugi parametar pojedinačan onda
    {recipročna procedura spajanja pojedinačan - manje od}
  inače ako je drugi definiran s „manje od“ onda
    {recipročna procedura spajanja manje od – više od}
```

U prethodno opisanom algoritmu nigdje nije naveden slučaj kad su i prvi i drugi parametar istog tipa i to „više od“ ili „manje od“ jer će u takvom slučaju uvijek jedan parametar biti sadržan u drugom.

6.6.3.6. cIPaddress: spajanje IP adresa

Spajanje IP zapisa radi se s pojedinačnim IP adresama, IP rasponom (mreža je tip raspona) te IP adrese s *wildcard* maskom. Procedura spajanja kad je jedan od zapisa IP adresa:

```
ako je prvi parametar IP adresa onda
  ako je drugi parametar IP adresa onda
```

```
    ako su vrijednosti iste onda
      ako je inverzija različita kod parametara onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako su vrijednosti različite onda
      ako su oba parametara s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako su oba parametara bez inverzije onda
        ako je jedan parametar za 1 veći od drugog onda
          prvi parametar postaje raspon s drugim;
          dozvoljeno uklanjanje drugog pravila;
        inače ako parametri u binarnom zapisu imaju samo
        jedan različit bit onda
          prvi parametar IP zapis s maskom kojoj su svi
          bitovi fiksni (1) osim zajedničkog;
          dozvoljeno uklanjanje drugog pravila;
  inače ako je drugi parametar IP raspon onda
    ako su oba parametra bez inverzije onda
      ako je prvi parametar rubni IP prostora, a drugi
      pokriva sve druge IP zapise onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je prvi parametar za 1 IP adresu ispred prvog
      elementa raspona onda
        prvi parametar postaje drugi parametar (raspon),
        kojemu je prvi element zamijenjen s originalnim prvim
        parametrom;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je prvi parametar za 1 IP adresu iza zadnjeg
      elementa raspona onda
        prvi parametar postaje drugi parametar (raspon),
        kojemu je zadnji element zamijenjen s originalnim
        prvim parametrom;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je samo prvi parametar s inverzijom onda
      ako je prvi parametar dio raspona onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je samo drugi parametar s inverzijom onda
      ako je prvi parametar jednak prvom u IP rasponu onda
        ako se raspon sastoji od 2 elementa onda
          prvi parametar postaje zadnja IP adresa iz raspona;
          dozvoljeno uklanjanje drugog pravila;
        inače ako se raspon sastoji od više elemenata onda
          prvi parametar postaje drugi parametar (raspon)
          kojemu se prvi element povećava za 1 IP adresu;
          dozvoljeno uklanjanje drugog pravila;
      inače ako je prvi parametar jednak zadnjem iz IP
      raspona onda
        ako se raspon sastoji od 2 elementa onda
          prvi parametar postaje prva IP adresa iz raspona;
          dozvoljeno uklanjanje drugog pravila;
```

```
        inače ako se raspon sastoji od više elemenata onda
          prvi parametar postaje drugi parametar (raspon)
          kojemu se zadnji element smanjuje za 1 IP adresu;
          dozvoljeno uklanjanje drugog pravila;
    inače ako su oba parametra s inverzijom onda
      ako se prvi parametar nalazi izvan raspona onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
  inače ako je drugi parametar IP adresa s wildcard maskom
  onda
    ako je samo prvi parametar s inverzijom onda
      ako je prvi parametar u obuhvatu „maske“ onda
        prvi parametar uklanja;
        dozvoljeno uklanjanje drugog pravila;
```

U nastavku slijedi procedura spajanja IP zapisa kad je jedan od zapisa IP raspon. U sklopu nje koriste se i prethodno opisane procedure `range_in_mask()` i `mask_in_range()` detaljnije opisane u poglavlju 6.6.2.6 te `range_mask_related()` detaljnije opisana u poglavlju 6.6.1.6.

```
ako je prvi parametar IP raspon onda
  ako je drugi parametar IP adresa onda
    {recipročna procedura već opisana}
  inače ako je drugi parametar IP raspon onda
    ako su rasponi identični onda
      ako je inverzija parametara različita onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je drugi raspon unutar prvog onda
      ako je samo drugi parametar s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je samo prvi parametar s inverzijom onda
        ako je parametrima jedan rubni element isti onda
          prvi parametar postaje smanjeni prvi raspon koji ne
          sadrži drugi raspon;
          dozvoljeno uklanjanje drugog pravila;
    inače ako je prvi raspon unutar drugog onda
      ako je samo prvi parametar s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je samo drugi parametar s inverzijom onda
        ako je parametrima jedan rubni element isti onda
          prvi parametar postaje smanjeni drugi raspon koji
          ne sadrži prvi raspon;
          dozvoljeno uklanjanje drugog pravila;
    inače ako se rasponi ne preklapaju onda
      ako su oba parametra bez inverzije onda
        ako su parametri susjedni rasponi onda
          prvi parametar se spaja s drugim parametrom u novi
          raspon;
          dozvoljeno uklanjanje drugog pravila;
        inače ako prvi parametar uključuje minimalnu, a drugi
```

```
        maksimalnu vrijednost, ili obrnuto onda
          prvi parametar postaje inverzni međuraspon;
          dozvoljeno uklanjanje drugog pravila;
        inače ako je su rasponi ujedno i mreže te njihove
        binarne zapisi imaju samo jedan različit bit onda
          prvi parametar postaje IP zapis s wildcard maskom
          niz koja je jednaka početnoj mrežnoj masci, samo
          joj je zajednički bit zamijenjen s 0;
          dozvoljeno uklanjanje drugog pravila;
      inače ako su oba parametra s inverzijom onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako se rasponi djelomično preklapaju onda
      ako su oba parametra bez inverzije onda
        prvi parametar se spaja s drugim parametrom u novi
        raspon;
        dozvoljeno uklanjanje drugog pravila;
      inače ako su oba parametra s inverzijom onda
        prvi parametar postaje zajednički međuraspon (s
        inverzijom);
        dozvoljeno uklanjanje drugog pravila;
      inače ako je samo prvi parametar s inverzijom onda
        prvi parametar postaje smanjeni raspon iz kojeg je
        uklonjen zajednički raspon s drugim parametrom;
        dozvoljeno uklanjanje drugog pravila;
      inače ako je samo drugi parametar s inverzijom onda
        prvi parametar postaje smanjeni drugi raspon iz kojeg
        je uklonjen zajednički raspon s prvim parametrom;
        dozvoljeno uklanjanje drugog pravila;
  inače ako je drugi parametar IP adresa s wildcard maskom
  onda
    ako su oba parametra bez inverzije onda
      ako raspon čini više od pola adresnog prostora i
      uključuje ili minimalnu ili maksimalnu IP adresu onda
        ako raspon uključuje 0.0.0.0 onda
          ako je range_in_mask() istinit za preostali raspon
          adresnog prostora (do 255.255.255.255) onda
            prvi parametar se uklanja;
            dozvoljeno uklanjanje drugog pravila;
        inače ako raspon uključuje 255.255.255.255 onda
          ako je range_in_mask() istinit za preostali raspon
          adresnog prostora (od 0.0.0.0) onda
            prvi parametar se uklanja;
            dozvoljeno uklanjanje drugog pravila;
    inače ako su oba parametra s inverzijom onda
      ako je range_mask_related() lažan onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
     inače ako je samo drugi parametar s inverzijom onda
      ako je mask_in_range() istinit onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je samo prvi parametar s inverzijom onda
```

```
      ako je range_in_mask() istinit onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
```

U nastavku je opisano utvrđivanje preklapanja kad su oba IP zapisa IP adrese s *wildcard* maskom:

```
ako je prvi parametar IP1 adresa s wildcard maskom W1 onda
  ako je drugi parametar IP2 adresa s wildcard maskom W2 onda
    ako su oba parametra bez inverzije onda
      ako parametri imaju istu wildcard masku, a IP adrese se
      razlikuju u samo jednom bitu onda
        prvi parametar modificira se tako da joj „razlikovni“
        bit u IP adresi i wildcard masci postaje 0;
        dozvoljeno uklanjanje drugog pravila;
    inače ako su oba parametra s inverzijom onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) != (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        prvi parametar se uklanja;
        dozvoljeno uklanjanje drugog pravila;
    inače ako je samo prvi parametar IP1 s inverzijom onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) = (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        ako je (W1 wildcard maska & W2 wildcard maska) = (W2
        wildcard maska) onda
          prvi parametar se uklanja;
          dozvoljeno uklanjanje drugog pravila;
    inače ako je samo drugi parametar IP1 s inverzijom onda
      ako je (IP1 adresa & W1 wildcard maska & W2 wildcard
      maska) = (IP2 adresa & W1 wildcard maska & W2 wildcard
      maska) onda
        ako je (W1 wildcard maska & W2 wildcard maska) = (W1
        wildcard maska) onda
          prvi parametar se uklanja;
          dozvoljeno uklanjanje drugog pravila;
```

Ako se IP raspon može zapisati kao mreža tada se taj IP zapis isto može promatrati kao *wildcard* maska kojoj su bitovi na početku postavljeni na 1, a bitovi na kraju na 0. Tako da ako je jedan parametar IP raspon koji je ujedno i mreža, a drugi IP adresa s *wildcard* maskom, za spajanje među IP zapisima koristi se zadnja opisana usporedba gdje su oba zapisa IP adrese s *wildcard* maske (izuzetak je jedino slučaj kad su oba parametra bez inverzije jer se tu nikako ne mogu spojiti mreža i IP adresa s *wildcard* maskom).

6.6.3.7. cDateTime: spajanje datuma i vremena

Algoritam spajanja datuma i vremena (inverzija nije dozvoljena kod Iptables alata):

```
ako prvi parametar nema definiran kraj onda
  ako drugi parametar nema definiran početak onda
    ako je početak prvog manji ili jednak kraju drugog
    parametra onda
      prvi parametar se uklanja;
      dozvoljeno uklanjanje drugog pravila;
  inače ako drugi parametar ima definiran početak i kraj onda
    ako je početak drugog parametra manji od početka prvog
    parametra te je kraj drugog parametra veći ili jednak
    početku prvog parametra
    onda
      prvom parametru početak se postavlja na vrijednost
      početka drugog parametra;
      dozvoljeno uklanjanje drugog pravila;
inače ako prvi parametar nema definiran početak onda
  ako drugi parametar nema definiran kraj onda
    ako je početak drugog manji ili jednak kraju prvog
    parametra onda
      prvi parametar se uklanja;
      dozvoljeno uklanjanje drugog pravila;
  inače ako drugi parametar ima definiran početak i kraj onda
    ako je početak drugog parametra manji od početka prvog
    parametra te je kraj drugog parametra veći ili jednak
    početku prvog parametra onda
      prvom parametru kraj se postavlja na vrijednost kraja
      drugog parametra;
      dozvoljeno uklanjanje drugog pravila;
inače ako prvi parametar ima definiran početak i kraj onda
  ako drugi parametar nema definiran početak onda
    ako je početak prvog parametra manji ili jednak kraju
    drugog parametra te je kraj prvog parametra manji od
    kraja drugog parametra onda
      prvom parametru uklanja se početak;
      dozvoljeno uklanjanje drugog pravila;
  inače ako drugi parametar nema definiran kraj onda
    ako je početak prvog parametra manji od početka drugog
    parametra te je kraj prvog parametra veći ili jednak
    početku drugog parametra onda
      prvom parametru uklanja se kraj;
      dozvoljeno uklanjanje drugog pravila;
  inače ako drugi parametar ima definiran početak i kraj onda
    ako je početak prvog parametra manji od početka drugog
    parametra te je kraj prvog parametra veći ili jednak
    početku drugog parametra onda
      prvom parametru kraj se postavlja na vrijednost kraja
      drugog parametra;
      dozvoljeno uklanjanje drugog pravila;
    inače ako je početak prvog parametra veći od početka
    drugog parametra te je početak prvog parametra manji ili
    jednak kraju drugog parametra onda
      prvom parametru početak se postavlja na vrijednost
      početka drugog parametra;
```

```
      dozvoljeno uklanjanje drugog pravila;
```

6.6.4. Utvrđivanje redundantnosti parametara

Jedan dio Firo optimizatora je i utvrđivanje redundantnosti parametara sigurnosnih pravila (6.1.7). Ti parametri su redundantni u drugom pravilu i kad bi akcije tih pravila bile identične pravila bi se mogla spojiti, a parametar po kojem se obavlja spajanje mogao bi se ukloniti.

Stoga se za utvrđivanje redundantnosti parametara koriste prethodno opisani algoritmi spajanja koji moraju rezultirati sljedećim akcijama:

```
prvi parametar se uklanja;
dozvoljeno uklanjanje drugog pravila;
```

Od svih implementiranih klasa jedino se kod parametra protokola ne radi uklanjanje jer je često protokol bitan za postojanje drugih parametara.

6.6.5. Utvrđivanje redundantnosti elemenata parametara

Optimizacijske tehnike uključuju i uklanjanje redundantnih elemenata iz parametara (6.1.8) što je trenutno realizirano kao uklanjanje portova iz niza portova (cPortNumber), kao i uklanjanje elemenata iz niza znakovnih vrijednosti (cString). Ti elementi mogu se ukloniti ukoliko su već sadržani u drugom pravilu.

6.6.5.1. cString: uklanjanje redundantnih elemenata niza znakovnih vrijednosti

Procedura utvrđivanja da li drugi znakovni parametar ima redundantne elemente uključuje modifikaciju drugog parametra:

```
ako su parametri multivalue onda
  ako su oba parametra neinvertirana onda
    ako prvi i drugi parametar imaju zajedničke elemente onda
      drugom parametru se uklanjaju elementi koji su ujedno
      elementi prvog parametra;
  inače ako su oba parametra invertirana onda
    ako je broj elemenata koji nisu sadržani u drugom
    parametru, a jesu u prvom parametru, manji od originalnog
    broja elemenata u drugom parametru onda
      drugi parametar postaje neinvertirani niz elemenata
      koji nisu sadržani u drugom parametru, a jesu u prvom;
  inače ako je samo prvi parametar invertiran onda
      drugom parametru se uklanjaju elementi koji nisu
      elementi prvog parametra;
  inače ako je samo drugi parametar invertiran onda
```

```
    ako je broj elemenata koji nisu sadržani u drugom
    parametru niti su sadržani u prvom parametru manji od
    originalnog broja elemenata u drugom parametru onda
      drugi parametar postaje neinvertirani niz elemenata
      koji nisu sadržani ni u prvom ni u drugom parametru;
```

6.6.5.2. cPortNumber: uklanjanje redundantnih portova iz niza portova

Procedura utvrđivanja ima li drugi parametar redundantne portove zahtijeva da drugi parametar bude niz, dok prvi parametar može biti i pojedinačan, raspon ili niz. Svaki put kad se broj elemenata niza smanjuje provjerava se ujedno je li ostao samo jedan element ili se preostali niz može konvertirati u raspon.

```
ako je prvi parametar pojedinačna vrijednost onda
  ako je drugi parametar niz vrijednosti onda
    ako su oba parametra bez inverzije onda
      ako je prvi parametar dio niza drugog parametra onda
        drugom parametru uklanja se element koji ima
        vrijednost jednaku prvom parametru;
    inače ako je samo drugi parametar s inverzijom onda
      ako je prvi parametar dio niza drugog parametra i
      akcija prvog pravila je blokirajuća onda
        drugom parametru uklanja se element koji ima
        vrijednost jednaku prvom parametru;
 inače ako je prvi parametar raspon onda
  ako je drugi parametar niz vrijednosti onda
    ako su oba parametra bez inverzije onda
      ako se barem jedan element niza nalazi u rasponu onda
        drugom parametru uklanjaju se oni elementi koji su
        sadržani u prvom parametru – rasponu;
    inače ako je samo prvi parametar s inverzijom onda
      ako se barem jedan element niza ne nalazi u rasponu
      onda
        drugom parametru uklanjaju se oni elementi koji nisu
        sadržani u prvom parametru – rasponu;
    inače ako je samo drugi parametar s inverzijom i akcija
    prvog pravila je blokirajuća onda
      ako se barem jedan element niza nalazi u rasponu onda
        drugom parametru uklanjaju se oni elementi koji su
        sadržani u prvom parametru – rasponu;
    inače ako su oba parametra s inverzijom i akcija prvog
    pravila je blokirajuća onda
      ako se barem jedan element niza ne nalazi u rasponu
      onda
        drugom parametru uklanjaju se oni elementi koji nisu
        sadržani u prvom parametru – rasponu;
inače ako je prvi parametar niz vrijednosti onda
  ako je drugi parametar niz vrijednosti onda
   ako su oba parametra bez inverzije onda
```

```
    ako se barem jedan element drugog niza nalazi u prvom
    nizu onda
      drugom parametru - nizu uklanjaju se oni elementi
      koji su sadržani u prvom parametru - nizu;
  inače ako je samo prvi parametar s inverzijom onda
    ako se barem jedan element drugog niza ne nalazi u
    prvom nizu onda
      drugom parametru - nizu uklanjaju se oni elementi
      koji nisu sadržani u prvom parametru - nizu;
  inače ako je samo drugi parametar s inverzijom i akcija
  prvog pravila je blokirajuća onda
    ako se barem jedan element drugog niza nalazi u prvom
    nizu onda
      drugom parametru - nizu uklanjaju se oni elementi
      koji su sadržani u prvom parametru - nizu;
  inače ako su oba parametra s inverzijom i akcija prvog
  pravila je blokirajuća onda
    ako se barem jedan element drugog niza ne nalazi u
    prvom nizu onda
      drugom parametru - nizu uklanjaju se oni elementi
      koji nisu sadržani u prvom parametru - nizu;
```

U slučaju kad je drugi parametar invertiran, smanjivanje broja elemenata parametra u pravilu povećava broj mogućih paketa koji će se procesuirati. Tako da je u takvim slučajevima smanjivanje broja elemenata dozvoljeno samo ako je akcija prvog pravila prekidajuća, tj. paketi s tim portovima nikad se neće pojaviti na drugom pravilu pa ih je moguće ukloniti iz inverznog (zabranjenog) niza.

6.7. Analiza učinkovitosti Firo optimizatora

Mjerenje učinkovitosti Firo optimizatora nije trivijalno budući da učinkovitost optimiranja ovisi o velikom broju faktora

- broj redundancija u originalnoj listi, što je najčešće u zavisnosti s brojem i kompleksnošću sigurnosnih pravila te iskustvom administratora,
- vrsta i učestalost mrežnog prometa te prosječan broj mrežnih konekcija,
- performanse računala na kojem se procesuiranje izvodi,
- verzija vatrozida.

Budući da su navedeni faktori različiti u različitim organizacijama, nije moguće definirati učinkovitost Firo egzaktnom brojkom. Stoga je za potrebe ovog rada provedena jednostavna analiza minimalnog vremena procesuiranja jedne sigurnosne naredbe dohvaćanjem male web datoteke (40 okteta) na Intel Genuine CPU 1.66GHz 1GB RAMa, pri čemu je zahtjev slan s istog računala na kojem se nalazi i web poslužitelj te se procesuiranje obavlja samo na INPUT lancu. Testiranje je provođeno na Iptables 1.4.4 vatrozidu te na starijoj verziji Iptables 1.2.11 vatrozida.

Dobiti vrijeme procesuiranja jedne naredbe nije trivijalno budući da HTTP zahtjev ne uključuje samo jedan paket, već više njih koje klijent i poslužitelj razmjenjuju, a određeno vrijeme potrebno je poslužitelju za slanje odgovora te klijentu za procesuiranje odgovora. Testiranje je uključivalo sljedeće korake: prvo se šalje velik broj zahtjeva (1000) kako bi se dobilo minimalno vrijeme potrebno poslužitelju za slanje odgovora, pri čemu na vatrozidu nema niti jedno definirano pravilo – $T_{min\ basic}$. Potom se redom testira niz od 1000, 10000 i 50000 pravila koja nisu zadovoljena, a ispred kojih se stavlja sljedeće pravilo:

```
-I INPUT 1 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Ono osigurava automatsko prihvaćanje svih paketa osim prvog poslanog (tzv. *stateful inspection firewall*) te se stoga procesuiranje - uspoređivanje paketa na definiranim pravilima obavlja samo za prvi poslani paket. Oduzimanjem minimalnog vremena potrebnog za procesuiranje niza od 1000 pravila $T_{min\ 1000}$, 10000 pravila $T_{min\ 10000}$ i 50000 pravila $T_{min\ 50000}$, od minimalnog vremena kad na vatrozidu nema definiranih pravila uspoređivanja $T_{min\ basic}$ dobije se vrijeme procesuiranja 1000 pravila, 10000 pravila i 50000 pravila. Dijeljenjem tih razlika s brojem pravila dobije se približno vrijeme procesuiranja po pojedinom pravilu. Iako, lista s 50000 pravila nije realistična ona se koristila samo kako bi se dobilo minimalno vrijeme procesuiranja jednog pravila.

Prve četiri metode optimiranja usmjerene su na uklanjanje sigurnosnih pravila – uklanjanje irelevantnih naredbi, anomalije u „sjeni iza“, anomalije u „sjeni ispred“ te uklanjanje zadnjih nepotrebnih pravila. Čak se i spajanjem dva sigurnosna pravila jedno pravilo modificira, a drugo uklanja.

Učinkovitost uklanjanja jednog sigurnosnog pravila ovisi o više faktora. Prvi faktor koji utječe na učinkovitost optimiranja jest koliko je pravilo složeno – što je složenije to će lista pravila biti učinkovitija kad se ono ukloni. A pravilo je složenije što ima veći broj parametara usporedbe te što mu je akcija zahtjevnija.

Primjer jednostavnog pravila s relativno malenim brojem (4) parametara usporedbe:

```
-A INPUT -p tcp -s 127.0.0.1/32 --sport 1000:65535 --dport 81
-j DROP
```

Primjer složenog pravila s relativno velikim brojem (12) parametara usporedbe:

```
-A INPUT -s 127.0.0.1/32 -i lo -p tcp -m tcp ! --tcp-option
11 ! --tcp-flags FIN,SYN,RST,PSH,ACK,URG NONE -m multiport --
dports 11,22,33,44,55,66,77,80,99,111,222,333 -m dscp ! --
dscp 0x00 -m mark ! --mark 0x15 -m mac ! --mac-source
```

```
AA:00:33:44:55:66 -m pkttype ! --pkt-type multicast -m length
--length 1:65535 -m state --state NEW,RELATED,ESTABLISHED -m
ttl --ttl-gt 20 -j DROP
```

Prosječna vremena uspoređivanja mrežnog paketa na jednom jednostavnom ili složenom pravilu za Iptables 1.2.11 i 1.4.4 prikazani su u sljedećoj tablici. Nažalost, novija verzija Iptables alata ne dozvoljava učitavanje velikog broja naredbi (50000) s velikim brojem parametara te ti podaci nisu raspoloživi.

Tablica 14: Minimalna vremena uspoređivanja mrežnih paketa na jednostavnim i složenim pravilima za Iptables 1.2.11 i 1.4.4

Broj pravila	**Minimalno vrijeme procesuiranja na pravilima INPUT lanca**			
	Ukupno vrijeme		**Vrijeme uspoređivanja po jednom pravilu**	
	Jednostavna pravila	Složena pravila	Jednostavna pravila	Složena pravila
Iptables 1.2.11				
1000	0.11 ms	0.20 ms	0.11 μs	0.20 μs
10000	0.88 ms	1.94 ms	0.08 μs	0.19 μs
50000	3.88 ms	9.76 ms	0.07 μs	0.20 μs
Iptables 1.4.4				
1000	0.05 ms	0.15 ms	0.05 μs	0.15 μs
10000	0.90 ms	2.0 ms	0.09 μs	0.20 μs
50000	4.09 ms	-	0.08 μs	-

Iz prethodne tablice vidljiva su slična vremena uspoređivanja za Iptables 1.2.11 te za 1.4.4. Iako starija verzija 1.2.9 može pohraniti 50000 složenih pravila, njihovo učitavanje traje čak 10ak minuta. Također, verzija Iptables 1.4.4 kod broja pravila do 1000 ima osjetno brže procesuiranje od verzije 1.2.11.

Iz podataka za Iptables 1.2.11 i 1.4.4 vidljivo je kako se jednostavnija pravila koja imaju samo 4 parametra usporedbe, uspoređuju u prosjeku 2-3 puta brže od složenih pravila koja imaju 12 parametara usporedbe (3 puta više od jednostavnih pravila). Na temelju toga potvrđena je korisnost optimiranja uklanjanjem redundantnih parametara, budući da se pravila s manjim brojem parametara efikasnije procesuiraju.

Kako bi se pokazala efikasnost uklanjanja redundantnih elemenata iz parametara korištene su dvije liste. Prva lista sadrži pravila s jednim definiranim ciljnim portom, dok druga lista sadrži pravila s nizom od 15 portova.

Primjer jednostavnog pravila s jednim definiranim ciljnim portom:

```
-A INPUT -p tcp -s 127.0.0.1/32 –dport 11 -j DROP
```

Primjer jednostavnog pravila s velikim brojem definiranih ciljnih portova:

```
-A INPUT -p tcp -s 127.0.0.1/32 -m multiport --dports
11,21,31,41,51,61,71,81,91,101,111,121,131,141,151 -j DROP
```

Tablica 15: Minimalna vremena uspoređivanja mrežnih paketa na pravilima s jednim i nizom portova

Broj pravila	**Minimalno vrijeme procesuiranja na pravilima INPUT lanca**			
	Ukupno vrijeme		**Vrijeme uspoređivanja po jednom pravilu**	
	Pravila s jednim portom	Pravila s nizom od 15 portova	Pravilo s jednim portom	Pravilo s nizom od 15 portova
Iptables 1.2.11				
1000	0.12 ms	0.14 ms	0.12 μs	0.14 μs
10000	0.89 ms	1.14 ms	0.09 μs	0.11 μs
50000	3.89 ms	5.20 ms	0.08 μs	0.10 μs
Iptables 1.4.4				
1000	0.03 ms	0.05 ms	0.03 μs	0.05 μs
10000	0.67 ms	1.32 ms	0.07 μs	0.13 μs
50000	22.21 ms	40 ms	0.44 μs	0.8 μs

Budući da je vrijeme procesuiranja pravila s jednim definiranim portom u pravilu jednako vremenu procesuiranja pravila s rasponom portova (od-do), vrijednosti vremena procesuiranja za raspon portova nisu priložene u prethodnoj tablici.

Kod Iptables 1.2.11 vatrozida pravila s nizom od 15 portova se u pravilu 0.02 μs duže procesuiraju što čini usporenje od 20-25%. Kod Iptables 1.4.4 pravila s nizom od 15 portova se osjetno duže procesuiraju s prosječnim usporenjem od 67-80%. Iz navedenog je vidljivo kako su pravila s manjim brojem elemenata u parametrima učinkovitija te da uklanjanje redundantnih elemenata iz parametara poboljšava efikasnost samog vatrozida.

Budući da Firo za razliku od drugih alata ne optimizira samo akcije prihvaćanja i odbacivanja, poseban dio analize optimiranja odnosi se na pravila koja imaju složeniju akciju – kao što je npr. logiranje zapisivanjem određenih podataka na sustav ili slanjem na udaljeni sustav.

Primjer jednostavnog logiranja bez akcijskih parametara:

```
-A INPUT -p tcp -s 127.0.0.1/32 --dport 80 –j LOG
```

Primjer složenog logiranja s akcijskim parametrima:

```
-A INPUT -p tcp -s 127.0.0.1/32 --dport 80 -j LOG --log-
prefix "INPUT local http packets" --log-level warning --log-
tcp-sequence --log-tcp-options --log-ip-options
```

Tablica 16: Minimalna vremena jednostavnog i složenog logiranja mrežnih paketa

Broj pravila	**Minimalno vrijeme logiranja na LOG pravilima INPUT lanca**			
	Ukupno vrijeme		**Prosječno vrijeme (na jednom pravilu)**	
	Jednostavno logiranje	Složeno logiranje	Jednostavno logiranje	Složeno logiranje
Iptables 1.2.11				
1000	0.15 s	0.24 s	147.22 μs	236.56 μs
10000	1.46 s	2.35 s	146.37 μs	235.44 μs
50000	7.36 s	11.82 s	147.27 μs	236.41 μs
Iptables 1.4.4				
1000	24.73 ms	40.82 ms	24.73 μs	40.82 μs
10000	234.1 ms	397.0 ms	23.41 μs	39.70 μs
50000	5.82 s	8.92 s	116.35 μs	178.35 μs

Iz prethodne tablice vidljivo je kako Iptables 1.4.4 ima bolje performanse po pravilu. U usporedbi s Iptables 1.2.11 logiranje s brojem pravila do 10000 je čak 6 puta brže, dok s porastom broja pravila taj odnos značajno pada pa iako 50000 pravila nije realistična veličina Iptables 1.4.4 je i dalje brži. A kad se pravila logiranja usporede sa složenim pravilima prihvaćanja ili odbacivanja prikazanim u tablici Tablica 14, vidljivo je su pravila s akcijama logiranja čak 700-1200 puta sporija za Iptables 1.2.11, tj. 100-200 puta sporija za Iptables 1.4.4. Iz toga je vidljiva korisnost jednog uklonjenog redundantnog pravila s akcijom logiranja koja je jednaka korisnosti uklanjanja nekoliko stotina ili čak tisuća pravila s akcijom prihvaćanja ili odbacivanja.

Osim što je Firo proširen za akcije logiranja, također optimizira i pravila s akcijama koje zahtijevaju mijenjanje mrežnog paketa kao što su npr. MARK i TOS koje postavljaju oznaku (engl. *mark*) tj. tip usluge (engl. *Type of Service*) na paket.

Primjer mijenjanja TOS vrijednosti:

```
-t mangle -A INPUT -p tcp -s 127.0.0.1/32 --dport 80 –j TOS
```

```
--set-tos 0x10
```

Primjer mijenjanja MARK vrijednosti:

```
-t mangle -A INPUT -p tcp -s 127.0.0.1/32 --dport 80 –j MARK
--set-mark 2
```

Tablica 17: Minimalna vremena modificiranja mrežnih paketa

Broj pravila	Minimalno vrijeme procesuiranja na pravilima INPUT lanca			
	Ukupno vrijeme		Prosječno vrijeme (na jednom pravilu)	
	TOS promjena	MARK oznaka	TOS promjena	MARK oznaka
Iptables 1.2.11				
1000	0.16 ms	0.15 ms	0.16 μs	0.15 μs
10000	1.11 ms	1.09 ms	0.11 μs	0.11 μs
50000	4.86 ms	4.74 ms	0.10 μs	0.09 μs
Iptables 1.4.4				
1000	0.11 ms	0.11 ms	0.11 μs	0.11 μs
10000	0.94 ms	0.96 ms	0.09 μs	0.10 μs
50000	29.26 ms	26.81 ms	0.59 μs	0.54 μs

U usporedbi s vremenima procesuiranja jednostavnih pravila s akcijama prihvaćanja ili odbacivanja iz tablice Tablica 14, vidljivo je kako je trajanje navedenih akcija jednako ili malo duže od jednostavnih pravila prihvaćanja ili odbacivanja (20-50%) čime je opravdano i optimiranje ovih tipova pravila.

6.8. Plan budućeg razvoja

U svojim počecima Firo je bio namijenjen optimiranju sigurnosnih pravila generiranih putem vatrozida razvijenog na Fakultetu elektrotehnike i računarstva (FER) u sklopu diplomskog rada [36] [38]:

Slika 14: Sučelje vatrozida razvijenog na FER-u

Budući da razvijeni vatrozid koristi Iptables alat za potrebe konfiguriranja sigurnosnih politika, Firo je brzo prenamijenjen te kao ulaz više ne koristi generirane skripte razvijenog vatrozida već izravno eksportirane Iptables naredbe, čime mu se proširuje mogućnost primjene.

U svom radu Firo trenutno u svakom koraku uspoređuje dva pravila. Budući da se većina povezanih pravila može spojiti, na posljetku je usporedba dvaju pravila uglavnom dovoljna jer se sve optimizacijske procedure ciklički ponavljaju dok god se niz pravila mijenja. Ipak, postoje parametri koji ne dozvoljavaju uvijek spajanje (npr. ukoliko se time dobiva niz portova duži od 15 kod Iptables alata). Stoga bi u novijim verzijama Firo alata trebalo omogućiti istovremeno uspoređivanje više od dva pravila istovremeno.

Optimiranje Firo alata usmjereno je na lanac naredbi. Pri tome se sigurnosna pravila iz različitih lanaca međusobno ne uspoređuju. Sljedeći korak u razvoju Firo optimizatora trebao bi biti povezivanje lanaca naredbi te uspoređivanje i optimiranje sigurnosnih pravila iz međusobno zavisnih lanaca. Takva vrsta uspoređivanja mogla bi se proširiti i na uspoređivanje sigurnosnih pravila iz većeg broja distribuiranih vatrozida.

Iako Firo omogućava korištenje i optimiranje pravila koji imaju niz portova (*multiport*), Firo trenutno ne omogućava korištenje i optimiranje pravila koja imaju niz raspona portova (npr. 2:10,20:30,40:50). Novija verzija Firo

programa trebala bi uključiti i takve nizove, a isto tako trebala bi se razviti nova programska klasa IP6address za potrebe IPv6 protokola.

Razvijeni program nema efikasnu vizualizaciju sigurnosnih politika kao npr. alat PolicyVis [58] što omogućava bolje razumijevanje definiranih politika, otkrivanje anomalija i konflikata. Trenutno je praćenje optimizacijskih koraka moguće kroz generirane datoteke. Nije nužno, ali u novijoj verziji Firo optimizatora poželjno bi bilo uključiti i vizualizacijsko grafičko sučelje koje bi administratorima olakšalo uočavanje anomalija te obavljenih optimizacija.

Nažalost, neke trenutne optimizacijske metode imaju uvjet da se između dva analizirana pravila ne mogu nalaziti druga pravila s neprekidajućom akcijom i akcijskim parametrima. Taj uvjet postoji jer ti akcijski parametri mogu modificirati mrežni paket ili neke druge elemente koji se uspoređuju u parametrima usporedbe. Takav uvjet prilično je restriktivan pa bi sljedeći korak isto morao biti klasifikacija akcijskih parametara te njihovo povezivanje s parametrima uspoređivanja. I, ako slijedno pravilo ne provjerava neki element koji među-pravilo mijenja, optimizacija bi bila dozvoljena.

Firo funkcionira kao statički optimizator, on optimizira sigurnosna pravila bez poznavanja šire situacije. Pri tome ne mijenja konačnu funkcionalnost vatrozida, paketi za koje je u originalnom nizu specificirano da moraju biti odbačeni nakon optimizacije, bit će odbačeni; paketi koji moraju biti prihvaćeni, bit će prihvaćeni i sl. Ukoliko bi se poznavala šira okolina, znalo bi se koji paketi nikad ni ne dolaze na vatrozid te koja su pravila irelevantna. Osim toga, uz poznavanje učestalosti pojave mrežnih paketa u određenim danima i vremenima, optimizator bi mogao generirati različite liste sigurnosnih pravila za različite dane i vremena. U tim listama, pravila s većom vjerojatnošću realizacije bi se nalazila bliže početku liste, a pravila s manjom vjerojatnošću realizacije bi se nalazila bliže kraju liste naredbi. Drugim riječima, Firo bi se realizirao kao dinamički optimizator.

Zadnji korak u unapređenju Firo programa bila bi njegova integracija s vatrozidima, pri čemu bi Firo pratio i analizirao trenutnu distribuciju mrežnog prometa, identificirao promjene u očekivanoj učestalosti te po potrebi mijenjao trenutno definirana sigurnosna pravila na vatrozidu.

Naravno, sve nadogradnje Firo optimizatora trebaju biti raspoložive na Firo službenim stranicama [39].

7. Zaključak

U organizacijama koje imaju nešto od navedenog: neiskusne administratore, velike ili kompleksno definirane sigurnosne politike, očita je korist optimizacijskih tehnika koje olakšavaju administratorima učinkovito upravljanje definiranim politikama. Čak i ako su administratori iskusni te nema čestih promjena kod odgovornih administratora, optimizacijske tehnike su korisne ukoliko vatrozidi posjeduju velike nizove sigurnosnih politika. Pri tome se misli na broj pravila od nekoliko stotina ili čak tisuća. Zato su u zadnjih 10ak godina objavljeni brojni radovi na temu optimiranja sigurnosnih pravila na vatrozidima, a u ovom radu predložen je i implementiran jedan novi mehanizam statičkog optimiranja sigurnosnih pravila – Firo (FIRewall Optimizer), optimizator prilagođen Iptables naredbama.

Prednost Firo optimizatora u odnosu na ostala rješenja je u tome što obuhvaća praktički neograničen broj parametara usporedbe, kao i neograničen broj mogućih akcija i akcijskih parametara. Naime, ostali radovi iz područja optimizacije vatrozida baziraju se na četiri, eventualno pet parametara te na dvije osnovne akcije prihvaćanja ili odbacivanja mrežnog prometa. Firo uzima u obzir sve parametre koji se mogu pronaći definirani među sigurnosnim pravilima, čak i ako oni nisu očekivani (u tom slučaju Firo njihovu vrijednost smatra znakovnom vrijednošću). Ipak, bolje je ako su parametri očekivani te konfigurirani kako bi se znalo radi li se o broju, rasponu, nizu, IP adresi, mreži i sl. Trenutno Firo ima konfigurirano preko 100 parametara uspoređivanja Iptables naredbi.

Još jedna prednost Firo optimizatora u odnosu na druga rješenja glede parametara je u mogućim vrijednostima parametara. Kod Firo optimizatora, zahvaljujući korištenju Iptables alata kao osnovice, dozvoljeno je korištenje inverznih vrijednosti parametara, raspona IP adresa, niza portova i drugih znakovnih elemenata. Korištenje takvih vrijednosti rezultira kompleksnijim operacijama koje se provode nad parametrima (utvrđivanje zavisnosti, utvrđivanje uključenosti, utvrđivanje sve-obuhvatnosti, spajanje).

Firo, za razliku od ostalih rješenja obuhvaća i druge akcije, a ne samo akcije prihvaćanja ili odbacivanja mrežnih paketa. Tako je moguće optimizirati i pravila koja su namijenjena preusmjeravanju (engl. NAT – *Network Address Translation*), logiranju mrežnog prometa, bilježenju stanja konekcije, statističke akcije, mijenjanju elemenata sadržanih u mrežnom paketu i dr. Tako da je optimizator primjenjiv i na *stateful* i na *stateless* vatrozide.

Osnovne optimizacije koje Firo optimizator provodi je uklanjanje redundantnih sigurnosnih pravila zbog raznih anomalija i spajanje sličnih pravila. Osim toga, za razliku od drugih poznatih optimizacijskih tehnika, program radi i tzv. manje optimizacije koje se odnose na mijenjanje

sigurnosnih pravila na temelju prethodnih pravila. Stoga se u pravilima uklanjaju određeni nepotrebni parametri ili elementi iz tih parametara.

Poseban dio Firo optimizatora je i rad s limitirajućim parametrima. Budući da je to područje dosta ovisno o potrebama administratora, ono je ostavljeno konfigurabilno da bi administrator mogao sam definirati na kakav način želi optimizirati pravila koja uključuju limitirajuće uvjete usporedbe.

Tijekom svog rada Firo samostalno uklanja, spaja i modificira sigurnosna pravila. Uslijed tih akcija Firo ne mijenja ukupnu sigurnosnu politiku i funkcionalnosti vatrozida. Oni ostaju isti, a zahvaljujući detaljnom izvještavanju koje uključuje generiranje izlaznih datoteka nakon svakog uspješnog koraka optimizacije, administrator može jednostavno pratiti koje akcije optimizacije su učinjene i koje anomalije su uklonjene.

Uz svoje prednosti, Firo također ima i svoje nedostatke. Prilikom rada program ne analizira više od dva sigurnosna pravila istovremeno. To je u slučaju kad se pravila mogu spajati zanemarivo, ali u slučajevima kada se pravila ne mogu zbog raznih ograničenja spojiti, to je ograničavajući faktor. Osim toga, Firo ne optimizira međusobno sigurnosna pravila ako se ona nalaze u različitim lancima ili čak na različitim distribuiranim vatrozidima. Iako je sustav izvještavanja prilično detaljan s opisom međukoraka, Firo trenutno nema grafičko sučelje niti sustav vizualizacije anomalija. I na kraju, Firo je trenutno samo statički optimizator, on ne uključuje u optimizacijske procese informacije o učestalosti mrežnog prometa.

Literatura

[1] Abedin M., Nessa S., Khan L., Al-Shaer E., Awad M. (2010): Analysis of Firewall Policy Rules Using Traffic Mining Techniques, International Journal of Internet Protocol Technology, Volume 5 Issue 1/2, str. 3-22.

[2] Abedin M., Nessa S., Khan L., Thuraisingham B. (2006): Detection and Resolution of Anomalies in Firewall Policy Rules, Lecture Notes in Computer Science, Volume 4127/2006, str. 15-29.

[3] Acharya S, Wang J, Ge Z., Znati T. F., Greenberg A. (2005): Traffic-Aware Firewall Optimization Strategies, University of Pittsbourgh, 2005.

[4] Acharya S., Abliz M., Mills B., Znati T. F. (2007): OPTWALL: A Hierarchical Traffic-Aware Firewall, Proceedings of the 14th Annual Network and Distributed System Security Symposium (NDSS'07), San Diego, SAD

[5] Acharya S., Wang J., Ge Z., Znati T., Greenberg A. (2006): Simulation Study of Firewalls to Aid Improved Performance, ANSS '06 Proceedings of the 39th annual Symposium on Simulation

[6] Al-Shaer E. S., Hamed H. H. (2002): Design and Implementation of Firewall Policy Advisor Tools, DePaul University, Chicago.

[7] Al-Shaer E. S., Hamed H. H. (2003): Firewall Policy Advisor for Anomaly Detection and Rules Editing, Proc. IEEE/IFIP 8th Int. Symp. Integrated Network Management (IM 2003), str 17-30.

[8] Al-Shaer E. S., Hamed H. H. (2004): Discovery of Policy Anomalies in Distributed Firewalls, Proceedings of the IEEE INFOCOM 2004, vol 23, no 1. Hong Kong, Kina, str. 2605-2626.

[9] Al-Shaer E. S., Hamed H. H. (2004): Modeling and Management of Firewall Policies, IEEE Transactions Network and Service Management, vol. 1. no 1; str 2-10.

[10] Al-Shaer E. S., Hamed H. H. (2006): Dynamic Rule-ordering Optimization for High-speed Firewall Filtering, ACM Symposium on InformAtion, Computer and Communications Security: Taipei, Taiwan, str. 21-24.

[11] Al-Shaer E. S., Hamed H. H., Boutaba R., Hasan M. (2005): Conflict classification and Analysis of Distributed Firewall policies, IEEE journal on selected areas in Communications, 23 (10), str. 2069-2084.

[12] AlgoSec Firewall Analyzer, http://www.algosec.com/, 1. veljače 2011.

[13] Bandara A. K., Kakas A. C., Lupu E. C., Russo A. (2009): Using Argumentation Logic for Firewall Configuration Management, Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management

[14] Basile C., Lioy A., Scozzi S., Vallini M. (2010): Ontology-based Security Policy Translation, Journal of Information Assurance and Security 5, str. 437-445

[15] Bhatt S., Horne W., Pato J., Rajagopalan S. R., Rao P. (2005): Model-based validation of enterprise access policies, http://www.hpl.hp.com/techreports/2005/HPL-2005-152R1.pdf, HP Laboratories, Princeton, 1. veljače 2011.

[16] Buttyán L., Pék G., Thong T. V. (2009): Consistency verification of stateful firewalls is not harder than the stateless case, Infocommunications Journal, vol. LXIV, no. 2009/2-3
[17] Capretta V., Stepien B., Felty A., Matwin S. (2007): Formal Correctness of Conflict Detection for Firewalls, Proceedings of the 2007 ACM workshop on Formal methods in security engineering, str. 22-30
[18] Chaure R. (2010): An Implementation of Anomaly Detection Mechanism for Centralized and Distributed Firewalls, International Journal of Computer Applications (0975 – 8887), Volume 7 br. 4, str. 5-8.
[19] Chaure R., Shandilya S. K. (2010): Firewall anamolies detection and removal techniques – a survey, International Journal on Emerging Technologies 1(1), str. 71-74
[20] Chomsiri T, Pornavalai C. (2006): Firewall Rules Analysis, Proceedings of the 2006 International Conference on Security and Management, SAM 2006, Las Vegas, SAD, str 213-219.
[21] Cohen E., Lund C. (2005): Packet classification in large ISPs: Design and evaluation of decision tree classifiers, ACM SIGMETRICS Performance Evaluation Review 33(1), str. 73–84.
[22] Condell M., Sanchez L. (2004): On the Deterministic Enforcement of Un-ordered Security Policies, BBN Technical Memorandum No. 1346
[23] Cuppens F., Cuppens-Boulahia N., Garcia-Alfaro J. (2005): Detection and Removal of Firewall Misconfiguration, In International conference on Communication, Network and Information Security (CNIS2005), Phoenix, SAD
[24] Doxygen, http://www.stack.nl/~dimitri/doxygen/, 1. ožujak 2011.
[25] Eronen P., Zitting J. (2001): An Expert System for Analyzing Firewall Rules, Proceedings of 6th Nordic Workshop on Secure IT-Systems (NordSec 2001), str. 100-107
[26] Fitzgerald W. M., Foley S. N., O'Foghlu M. (2007): Confident Firewall Policy Configuration Management using Description Logic, Twelfth Nordic Workshop on Secure IT Systems, Reykjavik, Island, October 11-12, 2007.
[27] Fulp E. W. (2005): Optimization of Network Firewall Policies Using Ordered Sets and Directed Acyclical Graphs, Proceedings of the IEEE Internet Management Conference (IM'05), Nice, Francuska, http://www.cs.wfu.edu/~fulp/Papers/ewflist.pdf, 1 veljače 2011.
[28] Golnabi K., Min R. K., Khan L., Al-Shaer E. S. (2006): Analysis of Firewall Policy Rules Using Data Mining Techniques, Network Operations and Management Symposium 2006, Vancouver, Kanada, str. 305-315.
[29] Gu Y., McCallum A., Towsley D. (2005): Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation, IMC '05 Proceedings of the 5th ACM SIGCOMM conference on Internet Measuremen, str 345-350.
[30] Hamed H., Al-Shaer E. (2006): On autonomic optimization of firewall policy organization, Journal of High Speed Networks 15, str. 209–227.

[31] Hamed H., El-Atawy A., Al-Shaer E. (2006): Adaptive Statistical Optimization Techniques for Firewall Packet Filtering, Proceedings of the 25th IEEE INFOCOM'06, Barcelona, Španjolska
[32] Hari A., Suri S., Parulkar G. (2002): Detecting and Resolving Packet Filter Conflicts, Proceedings of the 19th Joint Conference of the IEEE Computer and Communications Societies (INFOCOM00), str. 1203-1212
[33] Hari B, Suri S, Parulkar G. (2005): Detecting and Resolving Packet Filter Conflicts, Processing of 5th International Network Conference. Samos Islands, Grčka
[34] Hazelhusrt S. (1999): Algorithms for Analyzing Firewall and Router Access Lists, Technical Report TR-WitsCS-1999, Department of Computer Science, University of the Witwatersrand, Južnoafrièka Republika
[35] Iptables(8) – Linux man pages, http://linux.die.net/man/8/iptables, 1. veljače 2011.
[36] Katić T. (2004): Napredni Linux usmjerivač – vatrozid, Fakultet elektrotehnike i računarstva, diplomski rad, Zagreb
[37] Katić T., Pale P. (2007): Optimization of Firewall Rules. Proceedings of the 29th International Conference on Information Technology Interfaces; Cavtat, Hrvatska, str. 685-690.
[38] Katić T., Šikić M., Šikić K. (2005): Protecting and controlling Virtual LANs by Linux router-firewall. Proceedings of the 27th International Conference on Information Technology Interfaces; Cavtat, Hrvatska, str. 549-554.
[39] Katić T.: Firo optimizator, http://valeria.zesoi.fer.hr/~tkatic, 1. lipanj 2011.
[40] Khan B., Khan M. K., Mahmud M., Alghathbar K. S. (2010): Security Analysis of Firewall Rule Sets in Computer Networks, Emerging Security Information Systems and Technologies (SECURWARE)
[41] Khorram E., Mirzababaei S. M. (2005): Finding an Optimized Discriminate Function for Internet Application Recognition, The Second World Enformatika Conference, WEC'05, Istanbul, Turska, str. 160-163.
[42] Kim S., Lee H. (2010): Classifying Rules by In-out Traffic Direction to Avoid Security Policy Anomaly, KSII Transactions on Internet and Indormation Systems, Vol. 4. br 4., str. 671-690
[43] Kolehmainen A. (2007): Optimizing firewall performance, seminar, www.tml.tkk.fi/Publications/C/23/papers/Kolehmainen_final.pdf, 1. veljače 2011.
[44] Liu A. X. (2008): Formal Verification of Firewall Policies, ICC '08. IEEE International Conference on Communications Vol. 1, str. 1494-1498.
[45] Maselli G., Deri L., Suin S. (2003): Design and Implementation of an Anomaly Detection System: an Empirical Approach, Terena Networking Conference (TNC 03), Zagreb, Hrvatska
[46] Misherghi G., Yuan L., Su Z., Chuah C.-N., Chen H. (2008): A General Framework for Benchmarking Firewall Optimization Techniques, IEEE Transactions on Network and Service Management, Vol. 5, br. 4, str. 227-238

[47] Netfilter/iptables. 1999-2011; http://www.netfilter.org/documentation/, 1. veljače 2011.
[48] Northcutt S, Zeltser L, editors (2003): Inside Network Perimeter Security: The Definitive Guide to Firewalls, Virtual Private Networks (VPNs), Routers, and Intrusion Detection Systems. Indianapolis: New Riders Publishing
[49] Oliveira R. M., Lee S., Kim H. S. (2009): Automatic detection of firewall misconfigurations using firewall and network routing policies, [PFARM'09] IEEE DSN Workshop on Proactive Failure Avoidance, Recovery, and Maintenance (PFARM), Lisbon, Portugal
[50] Oskar Andreasson: IPTables tutorial 1.2.2, http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html, 1. veljače 2011.
[51] Rahmati M., Mirzababaei S.M. (2005): Data Mining on the Router Logs for Statistical Application Classification, The 2006 World Congress in Computer Science, Computer Engineering, and Applied Computing: The International Conference on Internet Computing (ICOMP 06), Las Vegas, SAD
[52] Rezvani M., Aryan R. (2009): Specification, Analysis and Resolution of Anomalies in Firewall Security Policies, World Applied Sciences Journal 7 (Special Issue of Computer & IT), str. 188-198.
[53] Salem O., Vaton S., Gravey A. (2007): A Novel Approach for Anomaly Detection over High-Speed Networks, European Conference on Computer Network Defense, Heraklion, Grčka
[54] Samak T., El-Atawy A., Al-Shaer E. (2007): FireCracker: A Framework for Inferring Firewall Policies using Smart Probing, Proceedings of the fifteenth IEEE International Conference on Network Protocols (ICNP'07), Peking, Kina, str. 294-303.
[55] Secure Passage (2010): FireMon – firewall cleanup whitepaper
[56] Tapdiya A., Fulp E. W.(2009): Towards Optimal Firewall Rule Ordering Utilizing Directed Acyclical Graphs, Proceedings of 18th International Conference on Computer Communications and Networks, San Francisco, SAD, str 1-6
[57] Tran T. (2008): Misconfiguration Analysis of Network Access Control Policies, magistarski rad, University of Waterloo, Ontario, Kanada
[58] Tran T., Al-Shaer E., Boutaba R. (2007); PolicyVis: Firewall Security Policy Visualization and Inspection, Proceedings of the 21st Large Installation System Administration Conference (LISA '07), Dallas, str 1-16.
[59] Wikipedia: Zipf-law, http://en.wikipedia.org/wiki/Zipf%27s_law, 15. srpanj 2011.
[60] Yuan L., Mai J., Su Z., Chen H., Chuan C.-N., Mohapatra P. (2006): FIREMAN: A Toolkit for FIREwall Modeling and ANalysis, Proceedings of the 2006 IEEE Symposium on Security & Privacy, str 199-213.
[61] Zaliva V. (2008): Firewall Policy Modeling, Analysis and Simulation: a Survey, http://www.crocodile.org/lord/fwpolicy.pdf, 1. veljače 2011.
[62] Zhang B., Al-Shaer E., Jagadeesan R., Riely J., Pitcher C. (2007): Specifications of A High-level Conflict-Free Firewall Policy Language for

Multi-domain Networks, Proceedings od the 12th ACM Symp. Access Control Models Technologies (SACMAT 2007), Francuska, str. 185-194.
[63] Ziegler R-L. (2002): Linux Firewalls. Indianapolis: New Riders Publishing

Prilog 1: Firo računalni program i korisnička dokumentacija

Firo računalni program zajedno s korisničkom dokumentacijom i primjerima optimiranja vatrozidnih pravila raspoloživ je na kompaktnom disku čiji je sadržaj naveden u nastavku:

- `FIRO-1.0.tar.gz` – izvorni kod Firo programa,
- `Win-FIRO-1.0.rar` – verzija programa prilagođena izravnom pokretanju na Windows operacijskim sustavima,
- `FIRO-documentation/` – direktorij s HTML i DOC dokumentacijom o izvornom kodu,
- `Test1/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test1.txt`) na kojima se mogu vidjeti rezultati optimiranja uklanjanjem irelevantnih pravila (rezultati vidljivi u `firewall_changes.txt` datoteci),
- `Test2/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test2.txt`) na kojima se mogu vidjeti rezultati uređivanja parametara usporedbe (rezultati vidljivi u `firewall_changes.txt` datoteci),
- `Test3/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test3.txt`) na kojima se mogu vidjeti rezultati optimiranja uklanjanjem anomalija u „sjeni iza“ (rezultati vidljivi u `firewall_changes.txt` datoteci),
- `Test4/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test4.txt`) na kojima se mogu vidjeti rezultati optimiranja uklanjanjem anomalija u „sjeni ispred“ (rezultati vidljivi u `firewall_changes.txt` datoteci),
- `Test5/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test5.txt`) na kojima se mogu vidjeti rezultati optimiranja uklanjanjem zadnjih nepotrebnih pravila (rezultati vidljivi u `firewall_changes.txt` datoteci),
- `Test6/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test6.txt`) na kojima se mogu vidjeti rezultati optimiranja spajanjem pravila (rezultati vidljivi u `firewall_changes.txt` datoteci),
- `Test7/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test7.txt`) na kojima se mogu vidjeti rezultati optimiranja zasnovanih na uklanjanju redundantnih parametara (rezultati vidljivi u `firewall_changes.txt` datoteci),
- `Test8/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test8.txt`) na kojima se mogu vidjeti rezultati optimiranja zasnovanih na uklanjanju redundantnih elemenata iz parametara (rezultati vidljivi u `firewall_changes.txt` datoteci),

- `Test9/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test9.txt`) na kojima se mogu vidjeti rezultati optimiranja zasnovanih na premještanju tzv. LOG pravila (rezultati vidljivi u `firewall_changes.txt` datoteci te u zadnjoj generiranoj verziji sigurnosnih pravila `firewall_1.txt`),
- `Test10/` – direktorij s primjerom ulaznih sigurnosnih pravila (`Test10.txt`) na kojima se mogu vidjeti rezultati optimiranja pravila koja imaju opciju limitiranja mrežnog prometa (rezultati vidljivi u `firewall_changes.txt` datoteci).

Sažetak

Naslov: Optimiranje sigurnosnih pravila vatrozida

Ključne riječi: Firo, statički optimizator vatrozid, anomalija, shadowed after anomalija, shadowed before anomalija, redundancije, spajanje pravila, LOG pravila, prekidajuće i neprekidajuće akcije, parametri usporedbe, akcijski parametri, Iptables

Ovaj rad prezentira Firo (***FIR****ewall* ***O****ptimizer*), statički optimizator sigurnosnih pravila prvenstveno baziran na Iptables vatrozidu, ali prilagodljiv i za druge tipove vatrozida koji sadrže sigurnosna pravila. U radu su opisane meni poznate metode optimiranja, od statički baziranih do dinamičkih koje prate karakteristike mrežnog prometa, kao i postojeća klasifikacija anomalija. Budući da Firo, za razliku od drugih tehnika, radi s mnogo većim brojem parametara uspoređivanja, a uz akcije prihvaćanja ili odbacivanja uvodi i druge tipove akcija (koje su proširive i s akcijskim parametrima), uz postojeću klasifikaciju anomalija definirani su i novi tipovi anomalija kao i nove tehnike optimizacije. Tako je u radu već poznata *shadowing* anomalija nazvana *shadowed after* te je proširena i na neprekidajuće akcije, a zatim je predstavljena *shadowed before* anomalija. Osim toga, predstavljene su i tzv. mikro-optimizacije koje su zasnovane na uklanjanju redundantnih parametara iz sigurnosnih pravila te na uklanjanju redundantnih elemenata iz parametara sigurnosnih pravila. Dio dokumenta posvećen je i optimiranju sigurnosnih pravila za logiranje i limitiranje mrežnog prometa. Budući da postojeći radovi pretpostavljaju uglavnom jednostavne parametre, u ovom radu su predstavljeni i algoritmi uspoređivanja i spajanja različitih složenih tipova parametara.

Summary

Title: Optimisation of firewall security rules

Keywords: Firo, firewall static optimizer, anomaly, shadowed after anomaly, shadowed before anomaly, redundancy, merge rules, LOG rules, breaking and not-breaking actions, parameters of comparison, action parameters, Iptables

This paper presents Firo (**FIR**ewall **O**ptimizer), static firewall rules optimizer primarily based on Iptables, adaptable for other types of firewalls which include rules. Paper contains optimisation methods known to author, both static and dynamic optimisation methods, as well as the existing classification of anomalies. Since Firo works with larger number of comparing parameters then existing methods, and it includes actions that don't only accept or reject network packets, this work defines new types of anomalies as well as new optimisation techniques. So, in this paper already known *shadowing* anomaly is extended to non-breaking actions and renamed to *shadowed after* anomaly; because new *shadowed before* anomaly is defined as well. New micro-optimizations, based on elimination of redundant parameters from rules and on elimination of redundant elements from parameters, are also presented. One part of this paper is dedicated to optimizing logging rules and rules which are limiting network traffic. Because existing papers assume simple parameters, this work presents algorithms for comparing and merging different types of complex parameters.

Životopis

Tihomir Katić rodio se 1981. godine u Karlovcu. Prvi dio djetinjstva proveo je u Slunju, a iz njega je 1991. pred srpskom agresijom prognan u Zagreb. U Zagrebu je pohađao XV. gimnaziju te 1999. maturirao. 2004. diplomirao je računarstvo na Fakultetu elektrotehnike i računarstva (FER) na temu „Napredni Linux usmjerivač - vatrozid“ o čemu objavljuje i rad na međunarodnoj znanstvenoj konferenciji ITI 2005 u Cavtatu. 2010. diplomirao je menadžment na Ekonomskom fakultetu u Zagrebu (EFZG) s temom „Upravljanje projektima razvoja računalnih sustava“.

Prvo radno iskustvo stekao je radeći na studentskim projektima 2001/02 godine na FERu iz područja Linux operacijskih sustava i računalne sigurnosti. Nakon stečene diplome 2004. zaposlio se u Multimedijalnim kompjuterima d.o.o. u Zagrebu gdje je uglavnom radio na razvoju web aplikacija. Od 2005. radi kao vanjski suradnik na FERu na projektima iz područja računalne sigurnosti i računalnog razvoja. Kao jedan od projekata iz područja računalne sigurnosti proizašao je i Firo - optimizator sigurnosnih pravila o kojem objavljuje rad na ITI 2007 u Cavtatu. Od 2009. radi u Altima d.o.o. na projektima računalnog razvoja i podrške za telekomunikacijske operatere.