Automatizacija odziva sigurnosnog sustava umreženih vatrozida

Cigula, Luka

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 364

AUTOMATIZACIJA ODZIVA SIGURNOSNOG SUSTAVA UMREŽENIH VATROZIDA

Luka Cigula

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 364

AUTOMATIZACIJA ODZIVA SIGURNOSNOG SUSTAVA UMREŽENIH VATROZIDA

Luka Cigula

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 364

Pristupnik:	**Luka Cigula (0036509659)**
Studij:	Računarstvo
Profil:	Znanost o mrežama
Mentor:	prof. dr. sc. Željko Ilić

Zadatak: **Automatizacija odziva sigurnosnog sustava umreženih vatrozida**

Opis zadatka:

U dinamičnom okruženju Interneta, gdje napadači, tj. neovlašteni korisnici mreže brzo mijenjaju IP adrese kako bi izbjegli otkrivanje, ključno je uspostaviti brz i automatski odgovor na potencijalne napade. Automatizacija je neophodna za trenutačnu implementaciju zaštitnih mehanizama prilikom napada malicioznih aktera. Napadači često koriste upravljačku infrastrukturu iza legitimnih ili kompromitiranih domena kako bi nastavili svoje napade nakon prvotne infiltracije u sustav. Stoga je imperativ razviti dinamičke sustave s automatiziranim odgovorima kako bi se učinkovito suprotstavili prijetnjama i zaštitili integritet sustava. Vaš je zadatak analizirati mrežne, sigurnosne i tehnološke preduvjete potrebne za implementaciju rješenja na maliciozan pokušaj upada u mrežu. Potrebno je implementirati i opisati način dinamičke implementacije novih politika vatrozida te integraciju vatrozida i sigurnosno operativnog centra kroz sustav upozorenja. Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Rok za predaju rada: 28. lipnja 2024.

Zahvaljujem kolegama iz tvrtke Combis na vatrozidima za simulaciju koje su mi stavili na raspolaganje te tehničkoj podršci. Zahvaljujem kolegama iz tvrtke Verso Altima na znanju i profesionalnom iskustvu koje su podijelili sa mnom. Konačno, veliko hvala prof. dr. sc. Željku Iliću na strpljenju i podršci na mojem akademskom putu posljednje tri godine.

Sadržaj

Uvod 5

1. Vatrozid 1

1.1. Uloga vatrozida 1

1.1.1. Napadi protiv kojih vatrozid pruža zaštitu 1

1.2. Povijesni razvoj i potreba 2

1.2.1. Prva generacija 3

1.2.2. Prva generacija 3

1.2.3. Druga generacija 3

1.2.4. Treća generacija 4

1.2.5. Četvrta generacija 5

1.3. Vatrozidi nove generacije 6

1.3.1. Vatrozidi sa stanjem 6

1.3.2. Dubinsko ispitivanje paketa 7

1.3.3. Virtualne privatne mreže 8

1.3.4. Kvaliteta usluge 10

1.3.5. SSL dešifriranje 12

1.4. Vatrozidi u topologiji mreže 14

1.4.1. Demilitarizirana zona 14

1.4.2. Topologije demilitariziranih zona 15

1.4.3. Raspodijeljeni vatrozidi 17

2. Korištene tehnologije 19

2.1. Elasticsearch 19

2.1.1. Arhitektura grozda 20

2.1.2. Arhitektura čuvanja podataka 21

2.1.3. Arhitektura baze podataka 21

2.1.4. Kibana ... 23
2.1.5. OpenSearch ... 24
2.2. ElastAlert2 ... 24
2.2.1. Način rada ... 25
2.2.2. Pravila ... 25
2.2.3. Upozorenja ... 26
2.3. VirusTotal ... 27
2.4. PAN-OS ... 27
2.4.1. Sustav upravljanja ... 28
3. Opis rješenja ... 30
3.1. Arhitektura rješenja ... 30
3.2. SIEM servisi ... 31
3.2.1. Provizioniranje ... 31
3.3. Implementacija programske logike ... 32
3.3.1. Generiranje dnevničkih zapisa ... 32
3.3.2. Obrada upozorenja ... 33
3.3.3. Generiranje *ElastAlert2* pravila ... 34
3.3.4. Kreacija pravila na vatrozidu ... 34
Zaključak ... 36
Literatura ... 37
Sažetak ... 42
Summary ... 43
Privitak ... 44

Uvod

U rastućem digitalnom okruženju, kibernetička sigurnost postala je fokus organizacija svih veličina. Sofisticiranost i učestalost kibernetičkih napada zahtijeva robusne i prilagodljive sigurnosne mjere. Vatrozidi, kao prva linija obrane, igraju ključnu ulogu u zaštiti infrastrukture organizacija. Međutim, ručno upravljanje i ažuriranje pravila vatrozida je dugotrajan proces, sklon pogreškama, posebno u dinamičnom okruženju prijetnji koje brzo evoluiraju. Ručno kreiranje i održavanje pravila vatrozida predstavlja izazove, poput rizika od ljudske pogreške, kašnjenja u odgovoru na nove prijetnje te neučinkovitost pri rukovanju velikom količinom pravila. Ovi izazovi zahtijevaju automatizirano rješenje koje može dinamički ažurirati pravila vatrozida na temelju informacija o prijetnjama u stvarnom vremenu. Unutar opsega ovog rada je razvijeno rješenje koje automatizira kreiranje i upravljanje pravilima vatrozida na *Palo Alto* vatrozidima. Integracijom sustava za upravljanje sigurnosnim informacijama i događajima pomoću *Elasticsearch* i *ElastAlert2* tehnologija te korištenjem platformi za obavještavanje o prijetnjama kao što je *VirusTotal*. Ovo istraživanje nastoji poboljšati brzinu i učinkovitost upravljanja vatrozidom. Opseg ovog rada uključuje dizajn i implementaciju simuliranog SIEM okruženja koje obrađuje zlonamjerne logove, dohvaća podatke s platformi za obavještavanje o prijetnjama te automatski generira pravila na vatrozidu. Ograničenja uključuju ograničenja simulacijskog okruženja i specifičnih API-ja korištenih za obavještavanje o prijetnjama. Predloženo rješenje integrira simulirani SIEM sustav pomoću *Elasticsearch* i *ElastAlert2* za obradu simuliranih zlonamjernih logova generiranih *Python3* programskim skriptama. Sustav dohvaća podatke o zlonamjernim IP adresama s platformi za obavještavanje o prijetnjama poput *VirusTotala*, koji se zatim koriste za obavještavanje korisnika putem email notifikacija. Najvažnije, ove informacije se koriste za automatizaciju kreiranja i ažuriranja pravila vatrozida na *Palo Alto* vatrozidima. Automatizacijom ovih procesa, rješenje nastoji smanjiti rizik od ljudske pogreške, poboljšati brzinu odgovora na prijetnje i povećati ukupnu učinkovitost upravljanja mrežnom sigurnošću. Kao rezultat, ovaj rad podrazumijeva automatizirano upravljanje pravilima vatrozida kroz podatke koje generira sustav za upozoravanje. Predloženo rješenje ima za cilj stvoriti dinamičan i responzivan obrambeni mehanizam protiv kibernetičkih prijetnji.

1. Vatrozid

Godišnje 343 milijuna ljudi postaje žrtvom kibernetičkih napada [1]. U administrativnim i logičkim napadima poput krađe identiteta, drugih vrsta prevara ili krađe podataka često se kao vektor napada koriste kompromitirani sustavi i mreže. Sustavi koji štite infrastrukturu organizacije od vanjskog Interneta nazivaju se vatrozidima (eng. *Firewall*). Provođenjem unaprijed definiranih politika kroz pravila, vatrozidi filtriraju promet te osiguravaju izolaciju mreže od kaotičnog i često malicioznog okruženja Interneta [2]. Ovo poglavlje fokusirat će se na pregled tehnologije vatrozida, povijesnog razvoja vatrozida te mogućnosti koje vatrozidi pružaju u zaštiti mreže.

1.1. Uloga vatrozida

Cilj vatrozida kao mrežne tehnologije je prije svega da zadovolji i osigura generalna načela sigurnosti – da zaštiti povjerljivost podataka u mreži, da očuva integritet mreže te da osigura dostupnost mreže i ostale infrastrukture. Povjerljivost čuva tako što ne dopušta prolazak povjerljivih podataka kompanije iz mreže prema malicioznim akterima. Integritet čuva kroz zabranu pristupa podacima i infrastrukturi malicioznim akterima te tako onemogućava mijenjanje konfiguracija infrastrukture ili promjenu samih podataka. Dostupnost osigurava filtriranjem i sprječavanjem velikog volumena prometa koji bi mogao ugroziti raspoloživost sustava.

1.1.1. Napadi protiv kojih vatrozid pruža zaštitu

Iako ni jedan sustav pa tako ni vatrozid ne može u potpunosti zaštiti mrežu od narušavanja osnovnih principa sigurnosti, vatrozid primarno pruža zaštitu od sljedećih vrsta napada: neovlaštenog pristupa mreži, napada uskraćivanja usluge te pokušaja izviđanja mreže.

Napadi neovlaštenog pristupa mreži su napadi u kojima maliciozni akter pokušava na neovlašten način dobiti pristup mreži i infrastrukturi sustava. Napad se najčešće izvršava kroz kompromitirane vjerodajnice ili greške u implementaciji ili konfiguraciji sustava.

Vatrozidi ovakav napad sprječavaju uglavnom kroz potpuno blokiranje prometa prema malicioznom akteru. Također je na vatrozidu moguće implementirati dodatne mjere sigurnosti prilikom spajanja poput višefaktorske autentikacije ili unaprijed blokirati adrese mreža za koje ne postoji razlog da se spajaju u mrežu organizacije.

Napad uskraćivanja usluge (eng. *Distributed Denial of Service - DDoS*) konstituira pokušaj slanja velikog volumena prometa na vanjske pristupne točke sustava. Sustav zahtjeve pokušava obraditi pri čemu ga veliki volumen zahtjeva počne „gušiti“, odnosno sustavu ponestaje resursa za obradu velikog volumena zahtjeva. U tom slučaju, sustav ne može obrađivati ni legitimne zahtjeve koji su mu upućeni, što često dovodi do parcijalnog ili kompletnog ispada sustava te nedostupnosti njegovih servisa. Ovakvi napadi su uglavnom distribuiranog tipa, odnosno zahtjevi se ne šalju s jedne točke ili od strane jednog aktera. Napadač koristi mrežu unaprijed kompromitiranih računala ili drugih uređaja (eng. *Botnet*) koji iz različitih mreža i različitih geografskih lokacija generiraju i šalju ogromne količine malicioznog prometa [3]. Vatrozidi ovakve napade sprječavaju naprednim algoritmima odbacivanja prepoznatih malicioznih paketa ili potpunim blokiranjem kombinacijom vanjskih pristupnih točki, tipa paketa i podataka o izvorištu.

Treća vrsta napada je napad izviđanjem. Prilikom ovakvog napada, napadač šalje maliciozne, najčešće deformirane pakete i upite te pokušava ispitati odgovor i odziv sustava. Vrlo čest napad u ovoj kategorije je skeniranje mrežnih vrata (eng. *Port*) u kojem napadač pokušava otkriti koje pristupne točke sustava su „otvorene“, odnosno koje točke odgovaraju na eventualni upit. Napadač također želi saznati i druge informacije o sustavi i mreži, poput vrste implementiranih uređaja, verzije programske podrške, tipa i verzija operacijskih sustava te bilo koje druge informacije koje u budućnosti može eksploatirati [4]. Iako ovakav napad uglavnom nije destruktivne prirode, on pruža temelj na kojem će se bazirati budući napadi te gotovo uvijek signalizira potencijalan destruktivan napad u budućnosti. Vatrozidi ovakve napade sprječavaju tehnikama skrivanja sustava i mreže, pri čemu napadač može vidjeti samo vanjske točke sustava koje su uglavnom dodatno zaštićene i često opstruirane od strane vatrozida.

1.2. Povijesni razvoj i potreba

Vatrozid u najširoj mogućoj definiciji podrazumijeva zid ili neku drugu fizičku prepreku koja štiti objekt ili prostor od plamena. Danas je pojam vatrozid gotovo sinoniman s logičkim

ili fizičkim uređajima u mreži koji su smješteni na vanjski perimetar mreže te štite sustav organizacije od malicioznih aktera ili drugih prijetnji mreži. Kako se kroz povijest kontinuirano javljala potreba za zaštitom sustava i resursa bilo koje organizacije tako se 1980-ih godina, u samim začecima modernih mreža i Interneta vrlo brzo javila potreba za zaštitom logičkih sustava.

1.2.1. Prva generacija

U prvim iteracijama, danas poznatim kao „Nultoj generaciji“, vatrozidi zapravo nisu bili jasno definirani uređaji ili logički servisi. Počeli su kao koncept segmentacije mreže i zaštite mreže kroz politike na usmjerivačima (eng. *Network Router*) koji bi mrežu logički dijelili na segmente između kojih nije bilo mogućnosti „curenja“ paketa. Odnosno, s obzirom na početke mrežnih protokola te njihovih nesavršenosti, ideja je bila onemogućiti protokolima da prilikom grešaka ili u nekim rubnim slučajevima „procure“ u segment mreže u kojem nisu trebali biti. Vrlo brzo ovaj se koncept proširenjima pretvara u „Prvu generaciju“ vatrozida. Ubrzanim rastom uređaja i aktera na Internetu 1990-ih godina, kibernetička sigurnost poprima sve veću razinu važnosti. Javlja se potreba za uvođenjem sigurnosnih politika i procedura koje bi onemogućile maliciozan ili drugi neovlašten pristup mreži.

1.2.2. Prva generacija

„Prva generacija“ vatrozida se fokusira na filtriranje paketa. Ideja je bila pregledavati pakete na mrežnim sučeljima, uglavnom do OSI sloja 4 (dakle TCP i UDP protokola) te s obzirom na sadržaj paketa isti propustiti ili odbaciti. Analogno, ovaj sustav je podsjećao na provjeru riječi u rečenici, gdje bi se riječ po potrebi odbacivala bez pregleda šireg konteksta u koji je stavljena. Problem ovakvog pristupa je pristup „bez stanja“, odnosno zbog limitacije u resursima uređaja (primarno brzini memorije i procesorskoj snazi), vatrozidi nisu mogli održavati kontekst paketa, poput primjerice toka u kojem je paket došao. Drugim riječima, svaki paket se filtrirao po setu unaprijed definiranih statičnih pravila te se pregledavao kao izoliran fenomen [5].

1.2.3. Druga generacija

Limitacije prve generacije su često uzrokovale probleme, poput odbacivanja paketa unaprijed uspostavljene legitimne veze ili pak s druge strane propuštanja paketa koji su

zasebno zadovoljavali pravila, ali su bili dio malicioznog toka. Česta eksploatacija ovog problema je uzrokovala razvoj vatrozida „sa stanjem“, koji se pojavljuju već ranih 2000-ih godina. Oni se danas smatraju „Drugom generacijom“ vatrozida. U arhitekturu ovakvog sustava je uvedena tablica stanja, u kojoj su se držali podaci o trenutno aktivnim vezama te posljedično ispitivanje vezano uz stanje. Tablica stanja je držala listu legitimnih i potencijalno malicioznih tokova te bi prilikom dolaska provjerila je li paket u postojećem toku te ga sukladno tome propuštala. Ukoliko nije u nekom od unaprijed zapamćenih tokova, paket bi se ispitivao unaprijed definiranom politikom te se s obzirom na rezultat ispitivanja upisivao ili kreirao novi tok u tablici stanja. Ovakav pristup je osiguravao poznavanje šireg konteksta u kojem je paket dolazio te se prilikom donošenja odluka o propuštanju koristila veća količina dostupnih informacija o paketu, osiguravajući bolje donošenje odluka [6].

1.2.4. Treća generacija

Povećanjem prometa prema web sjedištima, javila se potreba provjere i blokiranja prometa na aplikacijskom sloju. Razvijaju se sustavi poput aplikacijskih vatrozida koji uz antivirusne programe i posredničke poslužitelje ispituju sadržaje upita ili programa na aplikativnoj razini. Ovakvi sustavi su blokirali maliciozne programe ili upite, sprječavali neželjenu e-poštu. Također, implementiraju se i rješenja virtualnih privatnih mreža (eng. *Virtual Private Network - VPN*) kroz koje se zaposlenici i suradnici s udaljenih točaka mogu spajati direktno u mrežu same organizacije. Ovi svi servisi su činili ujedinjenu platformu za upravljanje prijetnjama (eng. *Unified Threat Management - UTM*). Problem koji nastaje je što je svaki od ovih servisa odvojen entitet te iako zajedno osiguravaju sustav i mrežu, ne mogu komunicirati i razmjenjivati saznanja, politike i podatke o prometu koji stiže. 2008. godine kompanija *Palo Alto Networks* izbacuje na tržište prvi vatrozid nove generacije (eng. *Next Generation Firewall*), čija je glavna karakteristika objedinjavanje svih prije navedenih tehnologija te njihova orkestracija. Ovim ujedno počinje i „Treća generacija“ vatrozida, koja je još uvijek široko prisutna na tržištu. Ova generacija vatrozida više ne ispituje pakete samo po adresama, protokolima ili drugim statičkim karakteristikama paketa već ispituje njihov širi kontekstu u vidu toka u kojem dolaze, aplikacija na koje se primjenjuju i konkretnog sadržaja koji u sebi nose. Vatrozidi sada uključuju i sustave za prevenciju i otkrivanja napada (eng. *Intrusion Detection System – IDS, Intrusion Prevention System – IPS*) koji ispituju takozvane „potpise“ paketa (definirane kao set karakteristika paketa) na velikom setu pravila te po potrebi odbacuju pakete. Danas su neki od poznatijih primjerice *Snort*, *Suricata* i *Zeek*.

Široka primjena enkripcije prometa kroz SSL i TLS protokole je uzrokovala potrebu za procesima dešifriranja u kojima bi se vatrozid koristio kao posrednik u komunikaciji pri čemu bi mu krajnje točke vjerovale i dopuštale „otvaranje“ paketa. Ovaj korak, iako često upitne moralnih i legalnih karakteristika je nužan za dubinsku provjeru paketa (eng. *Deep Packet Inspection*) kako bi se obuhvatile sve moguće prijetnje nekom sustavu ili mreži. S obzirom da je okruženje na Internetu sve kaotičnije i ubrzano postaje podložno vrlo čestim promjenama, sustavi statičnih pravila postaju pretjerano rigidni i često prespori u reakcijama na napade [7].

1.2.5. Četvrta generacija

Napadači konstantno mijenjaju svoje podatke i karakteristike te lako zaobilaze statična pravila. S druge pak strane, konstantno mijenjanje pravila i politika manualnim metodama nije skalabilno u velikim sustavima. Svi ovi problemi su ubrzali pristup automatiziranja vatrozida. Korištenjem dinamičkih skripti, integracije s drugim alatima koji prepoznaju napada i upade te korištenjem umjetne inteligencije, tržište je unazad zadnjih nekoliko godina počelo razvijati i prihvaćati vatrozide „Četvrte generacije“. Njih karakterizira prije svega široka i konzistentna uporaba umjetne inteligencije. Metode strojnog učenja prvo rade na prepoznavanju normalnog, ili „nultog“ stanja sustava te tako stvaraju referentni etalon. U slučajevima u kojim promet iskače iz referentnog etalona, automatizirano se kreiraju nove politike i pravila koje u stvarnom vremenu blokiraju nove prijetnje. Ovakav pristup je posebice potreban u sustavima s velikom količinom podataka i uređaja kao što su sustavi koji ekstenzivno implementiraju Internet stvari (eng. *Internet of Things - IoT*). Sustavi koji implementiraju metode strojnog učenja još uvijek nisu široko zastupljeni iz dva primarna razloga. Prvi je problem resursa, jer takvi sustavi zahtijevaju veliku količinu procesorske snage za konstantnu obradu velike količine podataka. Drugi problem stvaraju sami modeli strojnog učenja, kojima treba relativno dugo vrijeme za uspostavu etalona koji često ne mora biti ispravan. Pristup „najbolje moguće“ kod strojnog učenja često može rezultirati propustima u politikama ili pak blokiranjem ispravnog, standardnog prometa. Također, još uvijek zahtijevaju ručne intervencije prilikom uvođenja novih parametara u sustav. Primjerice, ako kompanija nakon uspostave etalona počinje poslovati s novim akterima, ovakav sustav će promet od strane istih, bez ručne intervencije, početi odbacivati [8].

1.3. Vatrozidi nove generacije

Vatrozidi nove generacije uz standardno ispitivanje paketa odrađuju još čitav niz akcija nad paketima i tokovima podataka kako bi osigurali dostatnu razinu sigurnosti sustava. Kako bi postigli najširi mogući kontekst o tipu prometa, vatrozidi nove generacije koriste čitav niz alata i servisa koji rezultiraju ispravnom implementacijom politika i pravila. Ne postoji jednoznačna definicija koja objedinjuje sve alate koji se koriste u vatrozidima nove generacije. Naime, sukladno proizvođaču, potrebama tržišta te samoj implementaciji unutar organizacije, vatrozidi nove generacije koriste odabrani podskup svih dostupnih alata. Ovo poglavlje će se fokusirati na one alate koji se gotovo uvijek koriste u implementacijama vatrozida nove generacije [9].

1.3.1. Vatrozidi sa stanjem

Prije svega, kao osnovu vatrozidi nove generacije kao svoj temelj koriste standardne vatrozide „sa stanjem“. Ovaj dio arhitekture ispituje pakete na OSI razinama 3 i 4, dakle u sferi IP, ICMP, UDP i TCP protokola. Vatrozid skuplja podatke o svakom paketu koji kroz njega prolazi te ga stavlja u širi kontekst sukladno njegovim podacima poput: IP adrese, protokola, identifikatora sesije ili bilo kojeg drugog podatka iz sadržaja paketa koji može odrediti kojoj vezi pripada konkretan paket. Arhitektura ovakvog vatrozida sastoji se od dvije glavne komponente, pri čemu je jedna tablica stanja, a druga automat koji primjenjuje zadane politike. Svakom paketu se prvo pokušava pridijeliti neka od veza iz tablice stanja te ukoliko je ista pronađena i zadovoljava trenutne politike, paket se propušta. U suprotnom, paket prolazi kroz ispitivanje sukladno pravilima te ukoliko ih zadovoljava, propušten je, a njegov je tok zapisan u tablicu stanja. Ako paket krši bilo koju od zadanih politika, odbačen je te je njegova sesija uvrštena u listu nedopuštenih. Ovakav pristup odvaja podatkovnu i kontrolnu ravan. Na podatkovnoj ravni se paketi samo ispituju na temelju tablice stanja te ukoliko su dio postojeće sesije samo ih se propušta. Tek u slučajevima u kojima se dalje primjenjuju pravila i politike paket odlazi u kontrolnu ravan. Najveća prednost ovakve arhitekture ravni je značajno poboljšanje performansi jer se ne ispituju politike na svakom paketu, već ih se dobar dio nakon prve provjere propušta. Druge prednosti uključuju finiju granulaciju prilikom implementacije pravila. S obzirom na to da promet ima jasan kontekst, administrator vatrozida može primjenjivati politike koje se ne odnose isključivo na sadržaj paketa, već i na prirodu prometa. Također, primjena pravila na promet omogućava

jednostavniji način upravljanja samim politikama jer se ne moraju uzimati u obzir sve moguće karakteristike paketa što značajno olakšava administraciju u dinamičnim okruženjima. No, s druge strane granulacija može predstavljati značajan izazov u administraciji kompleksnih sustava jer je potrebno uzeti u obzir velik broj mogućih veza i vrsta prometa, a greške u implementaciji politika mogu biti pogubne za brz i efikasan rad sustava jer postoji mogućnost odbacivanja legitimnih paketa. Takve greške se uglavnom rješavaju ručno pa se ovakav sustav u praksi ne može smatrati u potpunosti automatiziranim. Drugi problem ovakvog filtriranja je što operira na razinama 3 i 4 OSI modela, što znači da nema pristup aplikativnim dijelovima paketa. Aplikacijski sadržaj paketa je gotovo uvijek kriptiran te ga je sustav ne može čitati što omogućava napadaču da sakrije maliciozne namjere unutar kriptiranog tereta. Upravo ovaj nedostatak pokrivaju suplementarni dijelovi arhitekture vatrozida nove generacije [10].

1.3.2. Dubinsko ispitivanje paketa

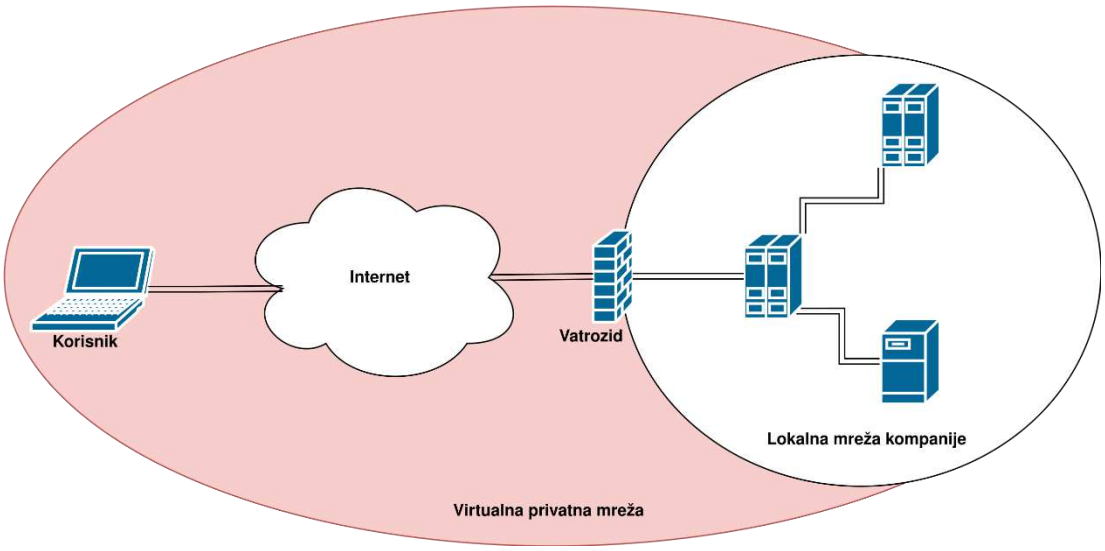
Primarni alat koji razdvaja moderne implementacije vatrozida nove generacije od vatrozida stanja je dubinsko ispitivanje paketa (eng. *Deep Packet Inspection – DPI*). Dubinsko ispitivanje paketa podrazumijeva analizu većeg broja podataka unutar samog paketa. Primjerice, standardni vatrozid stanja uglavnom ispituje podatke poput adresa pošiljatelja i primatelja, protokol i slične podatke koji se nalaze u zaglavlju paketa. U dubinskom ispitivanju se uz njih ispituju i dodatni metapodaci o paketu te njegov sadržaj. Sadržaj se može ispitivati direktno ukoliko promet nije šifriran. Nekoliko je načina na koji dubinska analiza paketa prepoznaje prirodu paketa. Prvi je metoda „potpisa“, u kojoj sustav može prepoznati podatke u sadržaju koji korespondiraju sa prometom koji je u njegovoj bazi otprije označen kao maliciozan. Određene vrste malicioznog koda ili prometa šalju karakterističan sadržaj koji je sustavu otprije poznat te prilikom prepoznavanja sustav može takav prometni tok odmah odbaciti i označiti kao nepoželjan [12]. Primarni problem ove metode je kontinuirana potreba za ažuriranjem baze potpisa kao reference prema kojoj se donose odluke. Iz tog razloga ova metoda nije uspješna u zaustavljanju novih ili nikad viđenih napada, poput napada „nultog dana“ (eng. *Zero Day Attack*). Kako bi se takvi napadi zaustavili, primjenjuje se metoda heuristike, odnosno svojevrsnog „pogađanja“. Sustavi dubinske analize prepoznaju određene uzorke u „ponašanju“ prometa te ukoliko se pojavi promet koji nije otprije poznat te zadovoljava neki od poznatih uzoraka, on se kao mjera dodatne sigurnosti odbacuje. Određeni tipovi maliciozne komunikacije, poput sustava za

naređivanje i upravljanje (eng. *Command & Control – C&C*) koji upravljaju unaprijed postavljenim malicioznim programima u mreži, slijede karakterističan uzorak slanja manjih paketa neuobičajenim protokolima (primjerice DNS protokolom). Ukoliko sustav prepozna ovakvo ponašanje, takav promet će biti odbačen. Također, ovaj pristup sprječava i iznošenje podataka jer će sustav prepoznati izlazak velike količine podataka iz mreže te odmah prekinuti tok. U dubinskoj analize se uvijek koristi načelo podrazumijevanog odbacivanja (eng. *Default Deny*) prilikom kojeg će sustav odbacivati sav promet, osim onog koji je eksplicitno dopušten u konfiguraciji [13].

1.3.3. Virtualne privatne mreže

Kako se vatrozidi gotovo uvijek nalaze na vanjskom rubu interne mreže organizacije, u funkciji poveznika (eng. *Gateway*) idealna su točka pristupa za virtualne privatne mreže (eng. *Virtual Private Network - VPN*). Virtualne privatne mreže su mehanizam za kreiranje sigurne veze između udaljenog sustava na Internetu i lokalne mreže u kojem udaljen akter ima pristup lokalno mreži kao da se u istoj fizički i nalazi [14]. S obzirom na opseg pristupa, virtualne privatne mreže dijele se u više kategorija.

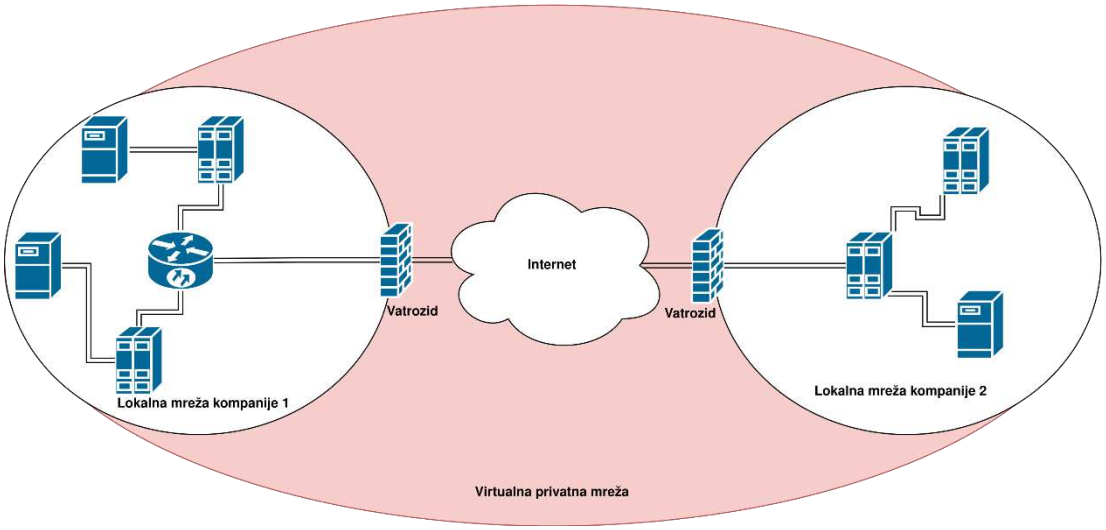
Prva vrsta pristupa je udaljen pristup (eng. *Remote access*) u kojem se jednom ili više korisnika ili uređaja dodjeljuje pristup lokalnoj mreži. Najčešće se koristi kao osiguravatelj mobilnosti zaposlenika koji s udaljenih lokacija imaju pristup internim servisima organizacije. Ovakav pristup je prikazan slikom (Slika 1.1).



Slika 1.1 Udaljen pristup preko virtualne privatne mreže

Druga vrsta je mjesto-do-mjesta (eng. *Site-to-Site*), pristup u kojem se osigurava konekcija između udaljenih mreža ili podmreža organizacije preko Interneta. Koristi se ukoliko organizacija ima udaljene i nepovezane lokalne mreže te želi logički simulirati jednu cjelovitu lokalnu mrežu prometom preko Interneta. Primjerice, područni uredi organizacije su kontinuirano preko Interneta spojeni u mrežu s centralnom lokalnom mrežom organizacije.

Konačno, treća vrsta pristupa je eksterno mjesto-do-mjesta, koje je gotovo identično drugoj metodi, no ovaj put se radi o spajanju mreža ili podmreža koje nužno ne pripadaju istoj organizaciji. Koristi se primjerice u okruženju u kojem dvije organizacije surađuju te postoji potreba za dugoročnim dijeljenjem internih resursa i servisa [15]. Princip druga dva pristupa je prikazan slikom (Slika 1.2).



Slika 1.2 Virtualna privatna mreža između dvije kompanije

Virtualne privatne mreže ekstenzivno koriste princip „prokopavanja“ tunela (eng. *Tunneling*) kao metodu zaštite moguće osjetljivih podataka koji putuju preko Interneta [16]. Virtualne privatne mreže se provode na slojevima 2 i 3 OSI modela, no vatrozidi uglavnom implementiraju rješenja sloja 3 pa će se ovaj rad fokusirati upravo na njih.

Najstariji i najrašireniji skup protokola implementacije VPN-a je *IPsec*. *IPsec* implementira dvije metode zaštite podataka, cjeloviti tunel (eng. *Full Tunnel*) u kojoj je čitav paket šifriran te „obavijen“ novim mrežnim paketom te transportna (eng. *Transport*) metoda u kojoj se samo teret šifrira, dok zaglavlja ostaju ista. Prednost *IPsec* modela je visoka razina međuoperabilnosti jer je široko raširen i često korišten, no kao glavna mana mu se pripisuje

visoka razina administracije zbog inherentne kompleksnosti te potrebe za upravljanjem ključevima i certifikatima [17].

SSTP (*Secure Socket Tunnel Protocol*) počiva nad SSL, odnosno TLS familiji protokola, koji su u širokoj uporabi s obzirom na raširenost HTTPS prometa. Također, koriste mrežna vrata 443 koja se uglavnom ne zatvaraju s obzirom da preko njih web sjedišta pružaju HTTPS pristup. Ovaj princip otvorenosti je najveća prednost ovog protokola jer ga je lako implementirati u rigidnim, zatvorenim sustavima. Kao nedostaci se navode nedostatak podrške za sustave koji ne implementiraju rješenja tvrtke *Microsoft* koja protokol održava. Također, SSL šifriranje nije izvorno namijenjeno ovakvoj vrsti prometa što može značajno usporiti prometne tokove preko SSTP-a [18].

Jedan od novijih protokola je *WireGuard*, protokol otvorenog koda koji implementira VPN preko UDP-a. *WireGuard* postaje sve popularniji kao alternativa ustaljenim protokolima jer ga je lako implementirati, koristi minimalno resursa te se pokazao izrazito brzim s obzirom na to da se kao projekt otvorenog koda može brzo prilagoditi i implementirati najnaprednije kriptografske metode. Nedostatak leži u potrebi za statičnim IP adresama, što povećava mogućnost ciljanih napada te nepovjerenju koje velike organizacije imaju prema alatu otvorenog koda naspram ustaljenih i dobro poznatih protokola [19].

OpenVPN je još jedna alternativa ustaljenim protokolima te koristi SSL, odnosno TLS kriptografske metode da zaštiti pakete koje može slati putem TCP-a ili UDP-a. Također je protokol otvorenog koda s vrlo robusnim metodama zaštite podataka. S obzirom na veliki broj mogućnosti, kompliciran je za uspostavu i administraciju te ne dolazi kao standardni paket na većini operacijskih sustava. Svaki od ovih protokola ima istu zadaću, osigurati povjerljivost, integritet i dostupnost podataka koji prolaze preko Interneta. Danas gotovo svi proizvođači vatrozida nude mogućnost implementacije bilo kojeg od VPN protokola te je izbor uglavnom određen potrebama i mogućnostima organizacija [20].

1.3.4. Kvaliteta usluge

Jedna od značajki vatrozida nove generacije je uspostava mehanizama osiguravanja kvalitete usluga (eng. *Quality of Service - QoS*), principa po kojem se određenom prometu daje svojevrstan prioritet u sferi iskorištavanja dostupnih mrežnih usluga kako bi se određene usluge neometano izvodile. Primjerice, usluge komunikacije preko IP protokola (eng. *Voice over IP – VoIP)* zahtijevaju znatno više mrežnih resursa kako bi se provodile neometano,

bez kašnjenja ili drugih vrsta smetnji. Brojni drugi servisi, poput IoT uređaja ovise o implementiranim politikama kvalitete usluge. Ispad veze, nedostavljanje paketa ili odgoda u dolasku mogu biti pogubni za rad određenih servisa te znatno narušiti usluge organizacije [21]. Ključni parametri kvalitete usluge su:

- Kašnjenje (eng. *Delay*) koje se mjeri kao količina vremena potrebna paketu da od izvorišta stigne na odredište.
- Propusnost (eng. *Bandwidth*) je mjerilo količine podataka koje može proći kroz određenu vezu u jedinici vremena.
- Gubitak paketa (eng. *Packet Loss*) je količina paketa koji su izgubljeni unutar komunikacije te rezultira gubitkom informacije.
- Podrhtavanje (eng. *Jitter*) opisuje nesrazmjer brzine dolaska paketa na odredište, najčešće zbog zagušenja mreže, što može dovesti do neispravnog redoslijeda dolaska paketa ili periodičkog zaustavljanja dolaska informacije.

Kako su vatrozidi najčešće vanjske pristupne točke neke kompanije te im je jedna od zadaća filtriranja i otkrivanja prometa, idealno su mjesto za implementaciju kvalitete usluge. Promet se prilikom prolaska kroz vatrozid klasificira, po mrežnim adresama i vratima, drugim metapodacima iz paketa ili heurističkim metodama. Nakon što je određen tip prometa, sukladno politikama na tom prometu se primjenjuju radnje koje osiguravaju zadanu kvalitetu usluge. Najčešće korištene metode osiguravanja kvalitete usluge su:

- Prioritizacija prometa kao proces u kojem se određenom prometu daje viši prioritet. Odnosno, prometni tok višeg prioriteta je propušten prije ostalih vrsta prometa kako bi se osigurala neometana usluga.
- Rezervacija resursa, implementirana RSVP protokolom pri čemu se određenoj vrsti prometa dodjeljuje veća količina mrežnih resursa.
- Implementacija reda (eng. *Queueing*), metoda kojom se promet prije prolaska kroz uređaj privremeno sprema u spremnik te se po unaprijed definiranoj metodi (primjerice po prioritetu) paketi puštaju dalje u mrežu.
- Označavanje paketa je postupak u kojem se paketima dodaju metapodaci koji označavaju prioritet prometa kako bi ostali uređaji u mreži znali kako dodijeliti resurse i brzinu prijenosa za određenu vrstu prometa.

Dobro implementiran sustav osiguranja kvalitete usluge osigurava brz, neometan promet, otporan na gubitke. Ključna prednost leži u kvalitetnom iskorištavanju dostupnih resursa, gdje svaka vrsta prometa ima maksimalnu iskoristivost sustava [22].

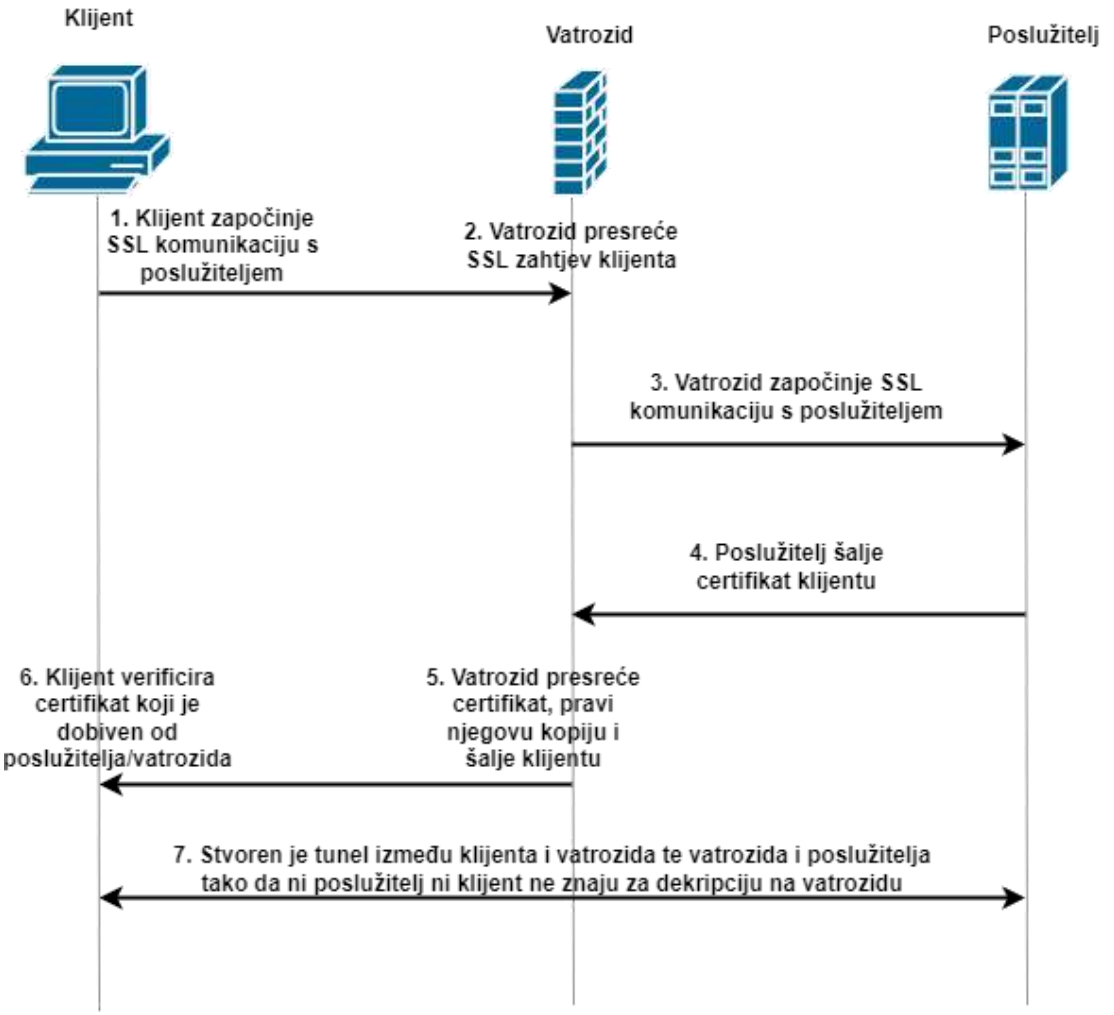
1.3.5. SSL dešifriranje

Preko 90% svog prometa na Internetu danas je šifrirano. Dapače, *Google* kao najveći pružatelj usluga pretrage web sjedišta, sjedišta koja ne implementiraju HTTPS enkripciju rangira niže, a svi moderni pretraživači izdaju upozorenja prilikom pristupa stranicama koje ne implementiraju HTTPS. Korištenjem HTTPS protokola koji koristi SSL ili napredniji TLS protokol, sav promet od i prema web sjedištima je šifriran. S obzirom da se ovaj postupak događa na slojevima 4,5,6 i 7 OSI modela, odnosno aplikativnim slojevima, vatrozid ne može dubinski ispitivati promet iznad sloja 3. Sav aplikativni promet, dakle svi podaci koje aplikacije šalju i primaju je vatrozidu nepoznat te posljedično na njega ne može primjenjivati zadane politike. Ako maliciozan akter sakrije svoje namjere ili maliciozan kod u šifriran promet, isti neće moći biti zaustavljen. Kako bi vatrozidi mogli ispitivati ovakav promet, uvode se metode otkrivanja šifriranog prometa [23].

Vatrozidi implementiraju rješenje otkrivanja korištenjem tehnike „čovjeka u sredini“ (eng. *Man in the Middle*), koja je inače vrlo čest alat napadača. Ova tehnika uključuje prihvat, obradu, a zatim i prosljeđivanje prometa između dva uređaja u mreži, bez da ijedan od uređaja za istu i zna. Ovaj pristup predstavlja prije svega legalne izazove jer osobe na koje utječe moraju biti informirane te dati svoj pristanak. Pristanak može biti opći ili se traži prilikom uspostave svake sjednice. Također, određene vrste prometa se ne smiju otkrivati, primjerice podaci o novčanim transakcijama. Drugi problem koji se stvara je uvođenje dodatnog procesiranja podataka [24]. Vatrozid svaki paket mora dešifrirati i obraditi, zatim ponovno šifrirati i slati dalje. Ovaj postupak može uvesti dodatno kašnjenje u mreži ili zagušenje u vidu iskorištenja resursa uređaja. Dešifriranje SSL prometa se odvija na dva načina, ovisno o smjeru uspostave konekcije.

Ukoliko se konekcija uspostavlja s izvorištem u lokalnoj mreži tada se radi o prosljeđivanju unaprijed (eng. *Forward Proxy*). Prilikom uspostave konekcije, klijent iz lokalne mreže šalje SSL zahtjev za uspostavu veze koji vatrozid presreće te s tim zahtjevom otvara sesiju prema vanjskom poslužitelju. Poslužitelj vraća svoj certifikat, koji vatrozid presreće te šalje

potpisanu kopiju certifikata klijentu. Nakon razmjene certifikata, razmjenjuju se kriptografski ključevi koje vatrozid presreće na isti način te ih čuva kako bi mogao dešifrirati promet. U stvarnosti, promet se odvija na način da vatrozid „glumi“ server klijentu, a klijent serveru. Server i klijent smatraju da se komunikacija odvija između njih bez posrednika, no svi paketi su primljeni dešifrirani, obrađeni, ponovno šifrirani te proslijeđeni od strane vatrozida. Ovaj proces je prikazan na slici (Slika 1.3).



Slika 1.3 Prikaz toka prometa kod dešifriranja prometa pri prosljeđivanju unaprijed

Drugi način je u primjeni kada SSL sesiju uspostavlja vanjski klijent prema poslužitelju u lokalnoj mreži. U tom slučaju se radi o ulaznom pregledu (eng. *Inbound Inspection*). U tom slučaju vatrozid sadrži certifikat i ključeve internog poslužitelja koji se prilikom uspostave sesije razmjene s vanjskim akterom. Odmah nakon razmjene vatrozid ima sve potrebne

ključeve te može nastaviti prosljeđivati promet te ga istovremeno dešifrirati. Ponovno, klijent i poslužitelj ne mogu znati da se promet dešifrira [25].

1.4. Vatrozidi u topologiji mreže

Najjednostavniji način pozicioniranja vatrozida u topologiji mreže je postavljanje na vanjski rub LAN mreže gdje vatrozid služi kao logička barijera između Interneta i interne mreže. Ovakav tradicionalan pristup se koristio u prvim iteracijama zaštite lokalnih mreža te je i dalje prikladan za male, kućne mreže ili mrežne sustave manjih organizacija. Prednost ove topologije je jednostavno održavanje i lako uvođenje strogih politika, jer sav promet između Interneta i lokalne mreže mora proći kroz vatrozid. Problem ovog pristupa je očigledan – upravo zato što sav promet mora proći kroz vatrozid, segmentacija vanjskih servisa organizacije u mreži postaje teška ili nemoguća [26].

1.4.1. Demilitarizirana zona

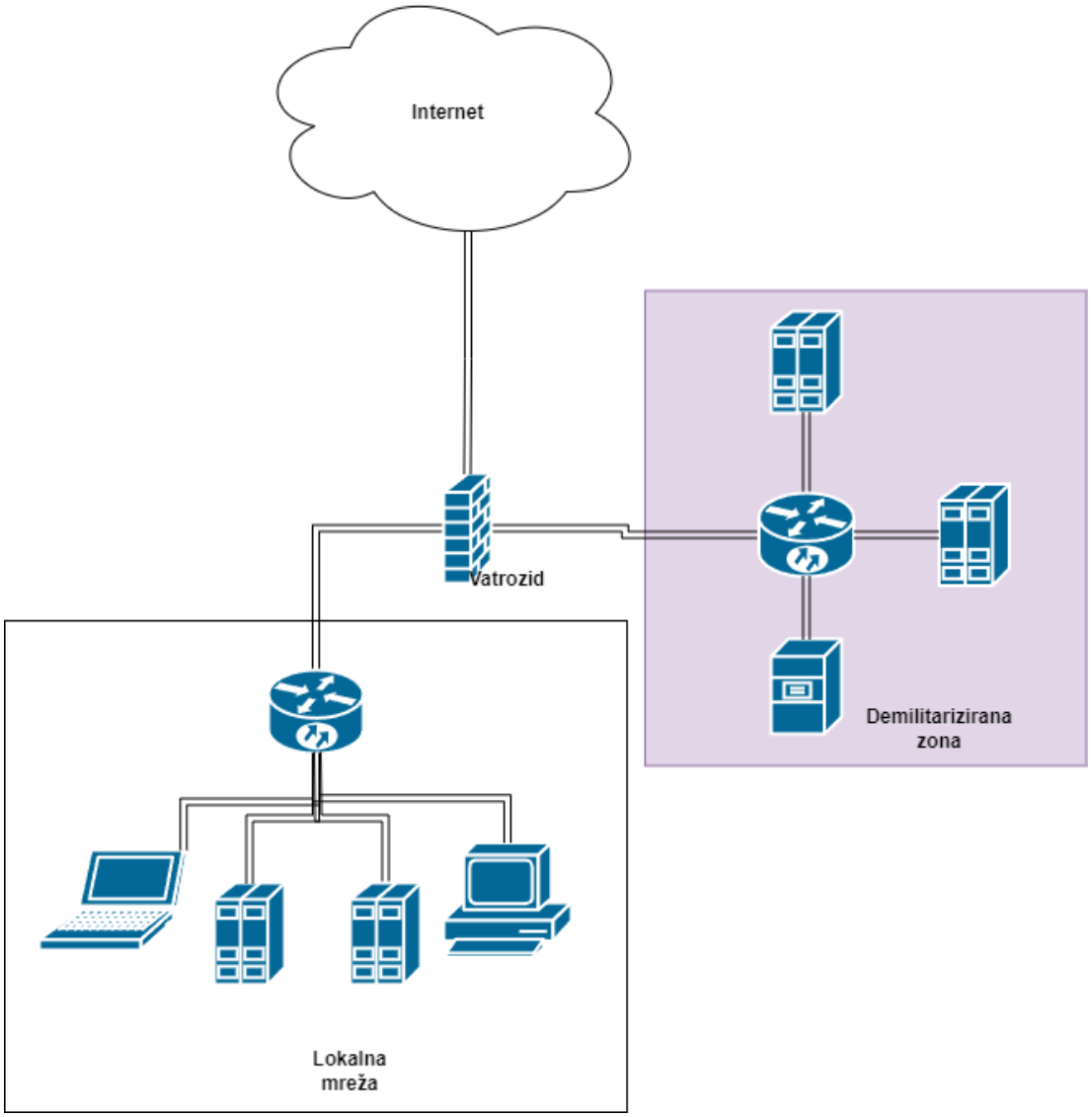
S obzirom na to da danas gotovo svaka organizacija koristi poslužitelje na kojima su aplikacije koje moraju biti dostupne na Internetu, stvara se potreba za dodatnom segmentacijom. Pristupom u kojem su vanjski servisi dostupni s Interneta, bez kompromitacije unutarnjih dijelova mreže. Upravo ova potreba je doprinijela razvoju demilitariziranih zona (eng. *Demilitarized Zone - DMZ*). Demilitarizirane zone su odvojeni segmenti mreže u kojima se nalaze servisi koji moraju biti dostupni s Interneta. Najčešće se radi o DNS poslužiteljima, servisima posrednika, FTP servisima, poslužiteljima e-pošte ili poslužiteljima web sjedišta organizacije. Kako su upravo ovi servisi izloženi Internetu, implicitno se prihvaća rizik da će isti biti podložni napadima [27]. Demilitarizirane zone se od napada štite na dva načina.

Prvi način je učvršćivanje poslužitelja (eng. *Hardening*), proces u kojem se identificiraju eventualne prijetnje poslužiteljima te se implementira rigorozna politika zaštita. Primjerice, osigurava se zadnja verzija programske podrške, zatvaraju se nekorištena mrežna vrata, uskraćuje se mogućnost elevacije prava te administratorski pristup, uklanjaju se nepotrebni servisi i aplikacije, kriptiraju se baze i ostali povjerljivi podaci te se konačno implementiraju sustavi nadzora. Bez obzira na sve ove mjere, priroda izloženih servisa je takva da određen dio podataka i proces mora biti dostupan vanjskim korisnicima te je tako dostupan i

potencijalnim napadačima [28]. Kako bi se demilitarizirana zona dodatno zaštitila uvodi se drugi način zaštite, a to su upravo vatrozidi.

1.4.2. Topologije demilitariziranih zona

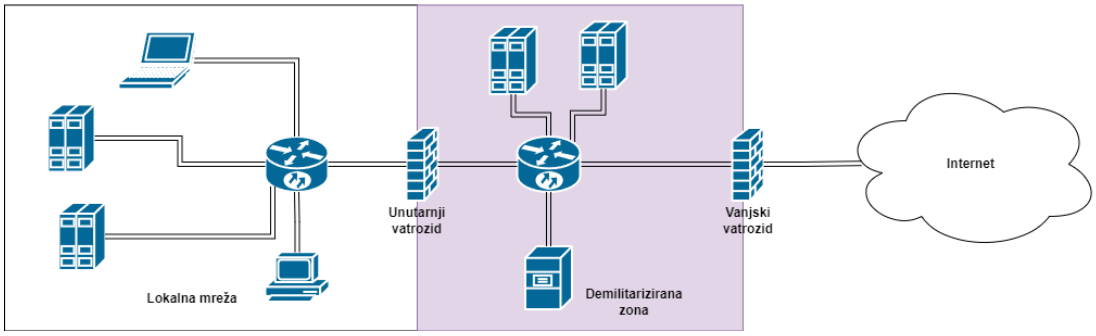
Dvije osnovne topologije demilitariziranih zona u odnosu spram vatrozida su „tronogi“ pristup te pristup dvojnih vatrozida. Vatrozidi u obje topologije pružaju dodatnu razinu segmentacije, odnosno izolacije demilitarizirane zone od interne mreže. Na taj način, ako servisi demilitarizirane zone i budu kompromitirani, ne utječu na ostatak interne mreže [29]. „Tronogi“ ili pristup provjerene podmreže (eng. *Screened Subnet*) je svojevrstan kompromis, u kojem je jedan vatrozid zadužen za sav promet demilitarizirane zone i interne mreže. Segmentacija je minimalna, s obzirom da je isključivo logička, sukladna konfiguraciji vatrozida. Princip je jednostavan, politike vatrozida su uvedene na način da se promet usmjerava sukladno odredištu te se po istom principu i filtrira. Ovakav pristup za prednost ima jednostavnost jer se i dalje koristi samo jedan vatrozid, no taj jedan vatrozid je također i jedinstvena točka ispada. Ukoliko on bude kompromitiran, kompromitirana je čitava mreža. Ovakva topologija se preporuča isključivo u slučajevima u kojima se u internoj mreži nalaze samo servisi koji trebaju pristup Internetu, a ne drže baze povjerljivih podataka ili druge aplikacije ključne za rad organizacije. „Tronogi“ pristup topologiji prikazan je slikom (Slika 1.4).



Slika 1.4 Topologija „tronogog“ pristupa

Druga i najčešće korištena vrsta topologije je pristup dvojnih vatrozida. U ovakvoj topologiji demilitarizirana zona stoji između dva vatrozida. Jednog vanjskog (često nazivanog perimetarskim) vatrozida koji je barijera između Interneta i demilitarizirane zone te jednog unutarnjeg (ili internog) vatrozida koji služi kao barijera između demilitarizirane zone i interne mreže. Ovakva vrsta segmentacije mreže prije svega pruža visoku razinu izolacije demilitarizirane zone i interne mreže jer svi upiti koji prolaze između dva segmenta moraju proći dodatnu filtraciju na internom vatrozidu. Također, uvodi se redundancija u zaštiti jer bilo kakav ispad vanjskog vatrozida ili kompromitacija demilitarizirane zone teško utječu na sigurnost interne mreže. Ovo je preferiran pristup te najčešće korišten u produkcijskim okruženjima. Nedostatak je što uvodi razinu dodatne kompleksnosti te povećane troškove

zbog većeg broja uređaja. Iako je to uglavnom slučaj, vatrozidi ne moraju segmentirati samo internu mrežu i demilitariziranu zonu, već mogu segmentirati i druge dijelove interne mreže kako bi se dodatno zaštitili osjetljivi interni servisi. Ovaj pristup prikazan je na slici (Slika 1.5). Dodatna segmentacija se uvodi sukladno potrebama i mogućnostima organizacije, najčešće odvajanjem servisa poput baza podataka dodatnim vatrozidom [30].



Slika 1.5 Topologija demilitarizirane zone između dva vatrozida

1.4.3. Raspodijeljeni vatrozidi

Tradicionalni vatrozidi provode filtriranje prometa na određenim ulaznim točkama mreže na temelju definiranih sigurnosnih politika. Učinkoviti su za male do srednje velike mreže, ali se suočavaju s izazovima zbog povećane povezanosti, složenih protokola i mreža. Tradicionalni vatrozidi ne mogu filtrirati interni promet koji ne vide, pretpostavljajući da su svi interni korisnici pouzdani. To dovodi do ranjivosti, posebno s protokolima koje je teško obraditi na razini vatrozida te pojavom sve veće uporabe neovlaštenih pristupnih točaka. Raspodijeljeni vatrozidi rješavaju ova ograničenja tako što centralno definiraju sigurnosne politike, ali ih provode na individualnim krajnjim točkama mreže. Te se politike mogu distribuirati pomoću te se provode na krajnjim točkama, što eliminira oslanjanje na topologiju mreže i smanjuje uska grla u performansama povezana s centraliziranim vatrozidima [31]. Ova arhitektura smanjuje ovisnost o mrežnoj topologiji i omogućava bolju zaštitu bez obzira na fizičku lokaciju uređaja. Ključni elementi raspodijeljenog vatrozida uključuju jezik za specifikaciju politika, sustave za upravljanje distribucijom politika, te *IPSec* za sigurnu komunikaciju. Politike se primjenjuju na raspodijeljene vatrozide kroz metodu „guranja“ (eng. *push*) gdje se politike postavljaju na vatrozide ili kroz metodu „povlačenja“ (eng. *pull*) gdje vatrozidi dohvaćaju konfiguraciju s neke udaljene administrativne točke [32]. Ovakav pristup ima mnogo prednosti. Prije svega, raspodijeljeni vatrozidi uklanjaju potrebu za specifičnim mrežnim rasporedom, što omogućava veću

fleksibilnost u topologiji mreže. Povećavaju otpornost jer više nema jedne točke na kojoj se može dogoditi kvar. Također, raspodjeljuju opterećenje na više mjesta te tako poboljšavaju performanse. Granulacija politika je izraženija te je lakše provoditi sigurnosne politike specifično za svaki uređaj ili korisnika. Kao najveći nedostatak se ističe kompleksnost upravljanja velikim brojem vatrozida te politika. Također, integracija velikog broja takvih servisa može biti opstruirana postojećim sustavom. Idealan model je kombinacija raspodijeljenog i tradicionalnog vatrozida kroz hibridni model vatrozida. Hibridni pristup raspodijeljenim vatrozidima kombinira prednosti tradicionalnih i raspodijeljenih vatrozida kako bi se postigla optimalna razina sigurnosti i učinkovitosti. Tradicionalni vatrozidi ostaju na ulaznim točkama mreže, kontrolirajući dolazni i odlazni promet. Oni služe kao prva linija obrane, osiguravajući osnovnu razinu sigurnosti i filtriranje neovlaštenog prometa. Krajnje točke, kao što su serveri i radne stanice, provode sigurnosne politike lokalno. Svaka krajnja točka može primijeniti specifične sigurnosne politike koje su distribuirane iz centraliziranog sustava za upravljanje. Kombinacija tradicionalnih i raspodijeljenih vatrozida omogućuje višeslojnu sigurnost. Tradicionalni vatrozidi štite mrežu na ulaznim točkama, dok raspodijeljeni vatrozidi pružaju dodatnu zaštitu na razini krajnjih točaka. U konačnici, hibridni pristup omogućuje organizacijama da iskoriste prednosti oba modela vatrozida, pružajući sveobuhvatnu i fleksibilnu sigurnosnu arhitekturu prilagođenu specifičnim potrebama i izazovima suvremenih mrežnih okruženja [33].

2. Korištene tehnologije

U ovom poglavlju bit će ukratko opisane arhitekture i mogućnosti tehnologija korištenih u implementaciji rješenja.

2.1. Elasticsearch

Elasticsearch je distribuirani alat za pretraživanje i analizu, koji predstavlja ključnu komponentu *Elastic Stacka* kao sustava za pohranu, obradu i analizu podataka. U kombinaciji s alatima kao što su *Logstash* i *Beats*, *Elasticsearch* omogućava efikasno prikupljanje, agregaciju i obogaćivanje podataka. Podaci se zatim pohranjuju unutar *Elasticsearch*a, gdje se odvija indeksiranje, pretraga i analiza. *Kibana,* kao pomoćni alat pruža korisničko sučelje za interaktivno istraživanje podataka, vizualizaciju, dijeljenje uvida, te upravljanje i nadzor cijelog sustava. *Elasticsearch* omogućava pretragu i analitiku u stvarnom vremenu za razne tipove podataka, bilo da se radi o strukturiranom ili nestrukturiranom tekstu, numeričkim podacima ili geoprostornim informacijama. Algoritmi unutar *Elasticsearch*a su optimizirani za brzo indeksiranje podataka, što omogućava korisnicima da provode složene upite i dobiju odgovore u realnom vremenu. Ova sposobnost čini *Elasticsearch* izuzetno korisnim u različitim primjenama, od pretraživanja weba do analize velikih skupova podataka u korporativnim okruženjima [34]. Jedna od ključnih prednosti *Elasticsearch*a je njegova skalabilnost. Kako organizacije nastavljaju generirati i prikupljati sve veće količine podataka, potreba za efikasnim analitičkim alatima postaje sve veća. *Elasticsearch* je dizajniran da raste zajedno s potrebama svojih korisnika, omogućavajući jednostavno dodavanje radnih čvorova i automatsko ravnomjerno raspoređivanje podataka i opterećenja preko svih dostupnih resursa. Ova raspodijeljena arhitektura ne samo da poboljšava performanse i dostupnost, već također povećava redundanciju te otpornost na kvarove. *Elasticsearch* također podržava različite dodatke koji proširuju njegove funkcionalnosti, uključujući sigurnosne dodatke, dodatke za praćenje performansi samog sustava te alate za upravljanje podacima. S obzirom na svoje svestrane sposobnosti i široku primjenjivost, *Elasticsearch* je postao nezamjenjiv alat za mnoge organizacije koje žele maksimalno iskoristiti svoje baze podataka [35].

2.1.1. Arhitektura grozda

Elasticsearch grozd (eng. *Cluster*) je skup povezanih čvorova (eng. *Node*) koji zajedno rade na pretraživanju i indeksiranju podataka. Grozd je identificiran jedinstvenim imenom koje svi čvorovi u grozdu dijele. Unutar grozda, podaci su raspodijeljeni među čvorovima, što omogućuje horizontalno skaliranje i povećava toleranciju na greške. Čvor je pojedinačna instanca *Elasticsearcha* koja pohranjuje podatke i sudjeluje u operacijama pretraživanja i indeksiranja. Svaki čvor pripada određenom grozdu i identificira se jedinstvenim imenom unutar tog grozda. *Elasticsearch* nudi nekoliko specijaliziranih tipova čvorova koji omogućuju optimizaciju performansi i pouzdanosti grozda. Ti tipovi uključuju čvorove *Master-eligible*, čvorove *Data*, čvorove *Ingest*, čvorove *Coordinating* i čvorove *Machine Learning*. Čvorovi *Master-eligible* su čvorovi koji mogu biti izabrani za glavni čvor. Glavni čvor je odgovoran za upravljanje operacijama nad cijelim grozdom kao što su kreiranje i brisanje indeksa, praćenje statusa grozda i upravljanje fragmentima. Glavnih čvorova može biti više te u tom slučaju, oni „glasaju“ prilikom donošenja odluka poput odluke o novom stanju čitavog grozda. Preporučuje se imati neparan broj čvorova *Master-eligible* kako bi se osigurala većina prilikom glasanja [36]. Čvorovi *Data* su čvorovi koji pohranjuju podatke i obavljaju operacije pretraživanja i indeksiranja. Čvorovi *Data* su ključni za performanse *Elasticsearch*a jer rukovode obradom podataka. Mogu biti konfigurirani s različitim količinama resursa kako bi se optimizirala njihova učinkovitost. Čvorovi *Ingest* su odgovorni za pred-obradu podataka prije nego što se oni pohrane u indeks. To uključuje procese poput parsiranja, obogaćivanja podataka i transformacije. *Ingest pipelines* tokovi omogućuju korisnicima definiranje nizova procesorskih operacija koje se primjenjuju na dokumente prije indeksiranja. Čvorovi *Coordinating* ne pohranjuju podatke i nisu uključeni u pretraživačke i indeksne operacije. Njihova primarna uloga je distribucija upita i agregacija rezultata iz različitih *data* čvorova. Čvorovi *Coordinating* pomažu smanjiti opterećenje na data čvorovima i poboljšati performanse pretraživanja. Čvorovi *Machine Learning* su specijalizirani čvorovi koji obavljaju zadatke vezane uz strojno učenje, kao što su detekcija anomalija i prediktivna analitika. Ovi čvorovi koriste resurse za procesiranje modela strojnog učenja, a njihova upotreba omogućuje napredne analitičke mogućnosti unutar *Elasticsearch* grozda [37].

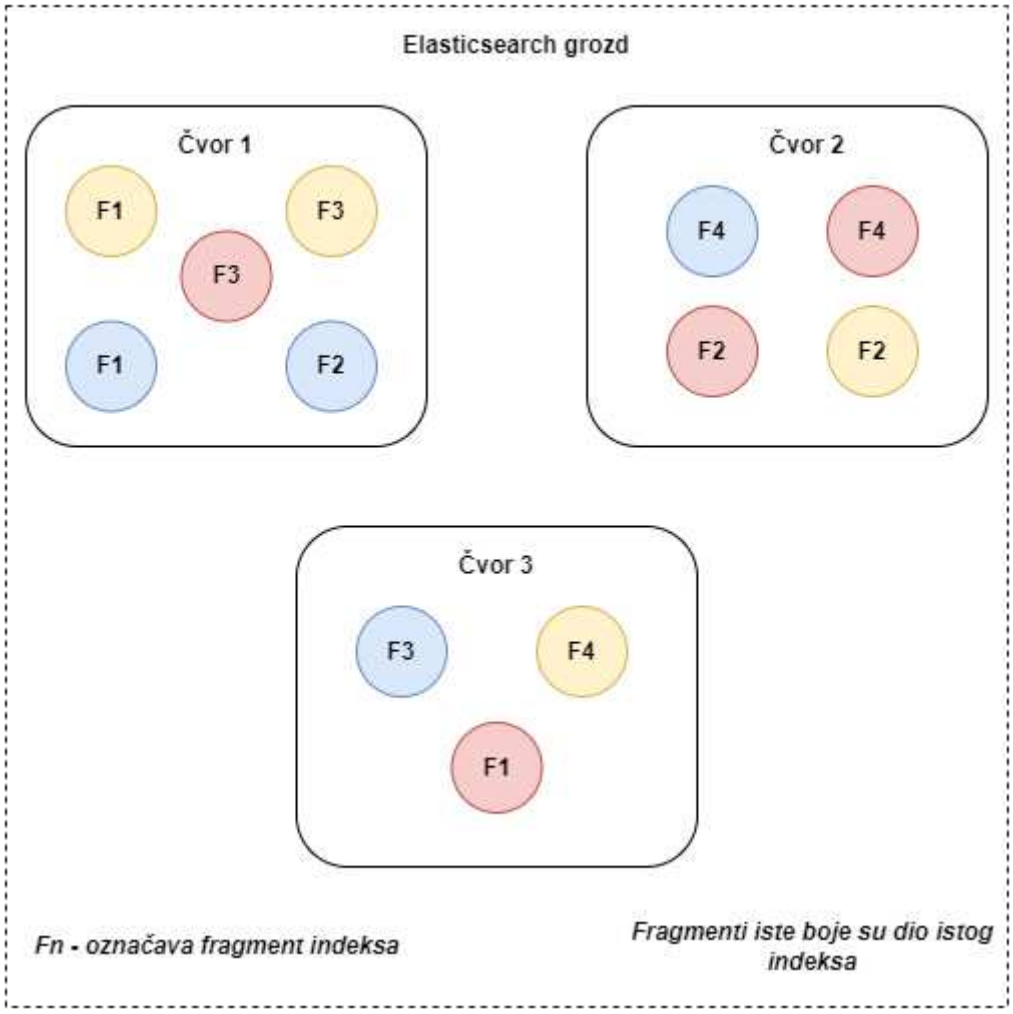
2.1.2. Arhitektura čuvanja podataka

S obzirom na to daje prije svega baza velike količine podataka, važno je osigurati minimalan utrošak resursa. *Elasticsearch* koristi različite tipove podatkovnih čvorova kako bi optimizirao pohranu i obradu podataka te osigurao učinkovito upravljanje resursima. Vrući čvorovi (eng. *Hot Node*) su dizajnirani za recentne podatke, koriste najviše resursa kako bi osigurali visoku brzinu i nisku latenciju. Idealni su za analitiku i pretraživanje u stvarnom vremenu, gdje su brzi odgovori ključni. Topli čvorovi (eng. *Warm Node*) pohranjuju podatke koji se rjeđe pretražuju. Omogućavaju balans između performansi i troškova, čuvajući podatke koji su još uvijek relevantni, ali manje kritični. Hladni čvorovi (eng. *Cold Node*) pohranjuju rijetko korištene podatke koristeći medije dugoročne pohrane poput velikih tvrdih diskova, s ciljem minimiziranja troškova i povećanja količine dostupnih podataka. Ovi podaci su dostupni, ali pristup može biti sporiji. Korištenjem tehnologije *Index Lifecycle Management (ILM)*, automatski premješta indekse kroz ove faze na temelju korisničkih politika, optimizirajući resurse i troškove. Ovakva arhitektura omogućava elastično skaliranje, ekonomično upravljanje podacima i visoku dostupnost [38].

2.1.3. Arhitektura baze podataka

Elasticsearch je izgrađen na otvorenoj biblioteci *Apache Lucene*. Jedan od njegovih ključnih aspekata je način na koji se podaci indeksiraju i pohranjuju, što omogućuje brze pretrage i analize. Indeksiranje je proces pripreme podataka za pretraživanje. U *Elasticsearchu*, podaci se organiziraju u indekse, koji su analogno najbliže relacijskim tablicama baza podataka u tradicionalnim sustavima upravljanja bazama podataka. Indeksi su označeni imenom koje je najčešće formata „*ime_indeksa-datum_kreacije*“ Svaki indeks sadrži jednu ili više zbirki dokumenata. Dokument je osnovna jedinica podataka pohranjena u *Elasticsearchu*. Dokumenti su pohranjeni u JSON formatu, što omogućava lako rukovanje podacima. Svaki dokument je označen jedinstvenim identifikacijskim brojem unutar indeksa i može sadržavati razna polja koja predstavljaju različite atribute podataka. Dokumenti u *Elasticsearchu* mogu biti vrlo heterogeni, što znači da dokumenti u istom indeksu ne moraju imati istu strukturu ili polja. Ovo omogućuje visok stupanj fleksibilnosti u pohranjivanju različitih tipova podataka u istom indeksu. Indeksi su podijeljeni u fragmente (eng. *Shard*) na način da se unutar svakog fragmenta nalazio dio dokumenata koji čine indeks. Svaki fragment je suštinski zaseban i neovisan indeks koji tvori vršni, logički indeks. Fragmenti

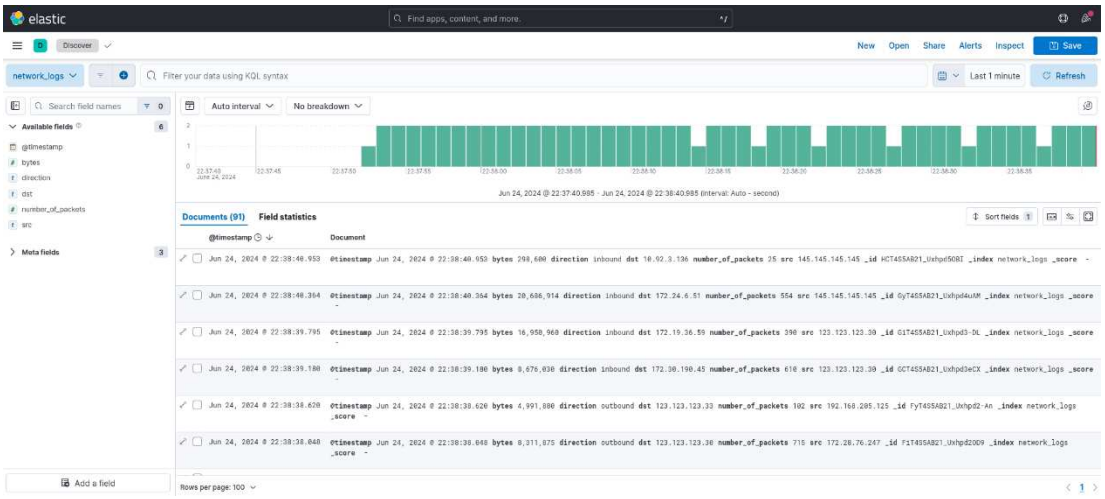
su podijeljeni između čvorova, što omogućava horizontalno skaliranje, distribuciju podataka i potrebnu redundantnost među različitim čvorovima unutar grozda [39]. Princip distribucije fragmenata prikazan je na slici (Slika 2.1). Kako bi pretraživanje baze bilo optimizirano, svi dokumenti se prije svega analiziraju te se njihov sadržaj pretvaraju u žetone (eng. *Token*). Ovaj postupak je najlakše predočiti rečenicom, u kojoj bi žetoni zapravo bili riječi [40]. Nakon što su žetoni stvoreni spremaju se u obrnuti indeks (eng. *Inverted Index*). Obrnuti indeks u svrhu jednostavnog predočavanja možemo zamisliti kao tablicu. U prvom stupcu se nalaze žetoni, a ostalih *n* stupaca je označeno identifikacijskim brojevima dokumenata. Ako žeton *i* postoji u dokumentu *j*, tada je polje *ij* označeno. Prilikom pretrage po nekom od žetona, mehanizam upita jednostavno prođe redom traženog žetona te brzo nalazi sve dokumente u kojima se nalazi. Ovakav sustav također omogućava lako rangiranje upita jer je broj dokumenata u kojem se neki žeton nalazi uvijek prisutan [41].



Slika 2.1 Prikaz fragmentacije indeksa u *Elasticsearchu*

2.1.4. Kibana

Kibana je alat za vizualizaciju i analizu podataka, razvijen kao dio *Elastic Stack-a* kako bi pružio jednostavan način za pregled i interakciju s podacima pohranjenim u *Elasticsearchu*. *Kibana* podržava *KQL* (*Kibana Query Language*), koji omogućuje pretraživanje i filtriranje podataka u indeksima. *KQL* je prije svega zamjena za *DSL* jezik koji se koristi za upite nad indeksima. *KQL* se prilikom izvođenja pretvara u *DSL*, ali je koncizniji i lakši za brzu uporabu. Jedna od ključnih značajki *Kibane* je mogućnost kreiranja prilagodljivih nadzornih ploča. Moguće je stvoriti različite vizualizacije kako bi se stvorili pregledni i informativne prikazi podataka. Način prikaza je širok, od osnovnih grafikona do složenih vizualizacija koje prikazuju trendove, anomalije i uzorke u podacima. *Kibana* također nudi podršku za geolokacijske podatke, omogućavajući korisnicima vizualizaciju podataka na kartama, što je korisno za analize koje uključuju geografske informacije. Osim toga, Kibana uključuje napredne analitičke mogućnosti, uključujući strojno učenje. Ove mogućnosti omogućavaju korisnicima automatsko prepoznavanje obrazaca i anomalija u podacima, pružajući dublji uvid i prediktivnu analizu. *Kibana* također podržava alarme i obavijesti, omogućavajući korisnicima postavljanje upozorenja koja se aktiviraju na temelju određenih uvjeta u podacima. Ova funkcionalnost je ključna za proaktivno upravljanje sustavima i brzo reagiranje na potencijalne probleme. Podržava i kontrolu pristupa na temelju uloga, omogućavajući administratorima da ograniče pristup određenim podacima i funkcionalnostima. *Kibana* se također može integrirati s vanjskim sustavima za upravljanje identitetom (poput LDAP-a), osiguravajući da samo ovlašteni korisnici mogu pristupiti osjetljivim informacijama [42]. Osnovni prikaz dokumenata u *Kibani* prikazan je na slici (Slika 2.2).



Slika 2.2 Prikaz *network_logs* indeksa u *Kibani*

2.1.5. OpenSearch

U kontekstu *Elasticsearcha,* važno je spomenuti i *OpenSearch.* Projekt *OpenSearch* nastao je kao odgovor *Amazona* na promjene u licenciranju *Elasticsearcha* i *Kibane*, svojim *Elastic License v2 (ELv2)* i *SSPL* licencama, kojima je *Elastic* ograničio njihovu slobodnu upotrebu. Stvorena je nova verzija pod *Apache 2.0* licencom, koja omogućuje korisnicima slobodno korištenje, prilagođavanje i daljnji razvoj softvera. *OpenSearch* osigurava alternativu *Elastic* proizvodima koji više nisu dostupni pod potpuno slobodnim licencama kroz otvoreni kod, nudeći slične funkcionalnosti i usklađenost s postojećim tehnologijama u ekosustavu. *OpenSearch* je račvanjem (eng. *fork*) koda iz verzije 7.10 *Elasticsearcha* nastao kao osiguranje da kompanije i privatni korisnici imaju potpuno otvorenu platformu za pretraživanje i analizu podataka koja nije ograničena promjenama u licenciranju koje bi mogle ograničiti njihovu upotrebu. Također, *OpenSearch* omogućuje korisnicima da nastave s prilagodbama i proširenjem funkcionalnosti bez brige o potencijalnim pravnim ili financijskim ograničenjima koja bi mogla proizaći iz promjene licence. Primjerice, moduli poput uporabe strojnog učenja, sigurnosni moduli te moduli koji prate stanje samog sustava više nisu dostupni u besplatnoj inačici *Elasticsearcha*, dok su kod *OpenSearcha* svi moduli dostupni i besplatni [43]. Ipak, *OpenSearch* i dalje kaska u performansama naspram *Elasticsearcha*, s obzirom da je njegov tim podrške znatno manji od onog *Elasticsearcha.* Tako je 76% brži prilikom pretraživanja, 58% brži kod sortiranja, 68% brži kod agregacija te koristi 37% manje podatkovnog prostora. Iako je razlika iz verzije u verziju sve manja, ekspertiza inženjera te zrelost samog rješenja još uvijek daju *Elasticsearchu* značajnu prednost [44].

2.2. ElastAlert2

ElastAlert2 je sustav otvorenog koda za obavještavanje dizajniran za praćenje podataka unutar grozda. Primarni cilj *ElastAlert2* sustava je otkrivanje anomalija, praćenje zapisa te slanje obavijesti sukladno zadanim pravilima putem različitih kanala. Nastao je kao nastavak i unaprjeđenje izvornog *ElastAlert* projekta razvijenog od strane organizacije *Yelp*. Danas je održavan od strane zajednice otvorenog koda kako bi nastavio pružati mogućnosti

obavještavanja, s obzirom na to da je modul za obavještavanje unutar *Elasticsearcha* prestao biti besplatan. Ova evolucija osigurala je da *ElastAlert2* ostane kompatibilan s najnovijim verzijama *Elasticsearcha* te da nastavi zadovoljavati rastuće zahtjeve modernog praćenja i obavještavanja [45].

2.2.1. Način rada

ElastAlert2 može se pokretati na kontejneriziran način poput *Docker* kontejnera ili na orkestriran način kroz tehnologiju *Kubernetes*. Također ga je moguće pokretati kao paket programskog jezika *Python3* [46].

Način rada definiraju pravila napisana u jeziku *YAML*. Sastoje se od metapodataka koji definiraju osnovne podatke o pravilu poput njegove aktivnosti ili imena. Konfiguracijskog dijela koji definira način na koji se pravilo povezuje s grozdom te drugim sustavima ili servisima s kojih se podaci prihvaćaju ili šalju. Ključan dio je definiranje rada pravila, u kojem se zadaje promatrani indeks te logika koju podaci moraju zadovoljiti da bi se pravilo aktiviralo. Konačno, sustav upozorenja definira način na koji će se korisnik upozoriti ukoliko se pravilo aktivira. Moguće je definirati i globalnu konfiguraciju koja će se primjenjivati na svako napisano pravilo, osim ako se u pravilo eksplicitno ne stvori konflikt s globalnom konfiguracijom. Tada će sustav primjenjivati logiku pravila. *ElastAlert2* u nekom unaprijed definiranom vremenskom razdoblju preko API-a ispituje vrijednosti unutar polja svih dokumenata u zadanim indeksima. Ukoliko se vrijednosti podudaraju s definiranom logikom pravila, tada će se pravilo aktivirati te odraditi unaprijed zadanu akciju [47].

2.2.2. Pravila

Konfiguracijski dio pravila definira koji se uvjeti moraju zadovoljiti kako bi se pravilo aktiviralo. Pravila filtriraju podatke s obzirom na tip definiran u konfiguraciji u polju „type:“. Tipovi pravila su:

- „Any“: Primjeni na sve. Drugim riječima, čim se unutar definiranih polja pojavi zadana vrijednost, aktivira pravilo.
- „Blacklist“: Aktivira pravilo ukoliko se u zadanom polju pojavi vrijednost navedena u „crnoj listi“.

- „Whitelist“: Aktivira pravilo ukoliko se u zadanom polju ne pojavi vrijednost navedena u „bijeloj listi“.
- „Change“: Aktivira pravilo ukoliko se u zadanom polju u zadanom vremenskom razdoblju pojavi vrijednost koja odudara od zadanih vrijednosti. Nakon što vremensko razdoblje prođe, zaboravi zadane vrijednosti.
- „Frequency“: Aktivira pravilo čim se u zadanom vremenskom razdoblju vrijednost pojavi više od dozvoljenog broja puta.
- „Spike“: Promatra neko vremensko razdoblje te frekvenciju pojave vrijednosti. Ukoliko u trenutno razdoblju pojava vrijednosti odudara od prijašnjih vremenskih okvira, aktivira pravilo.
- „Flatline“: Aktivira pravilo ako se u nekom vremenskom razdoblju određena vrijednost ne pojavi zadan broj puta.
- „New“: Aktivira pravilo ukoliko se u definiranom polju nađe prije neviđena, nepoznata vrijednost. Ta se vrijednost zatim dodaje u poznate vrijednosti [48].

2.2.3. Upozorenja

Kada se pravila aktiviraju, najčešći rezultat je generiranje upozorenja koje se može slati korisniku na različite načine. Neke od popularnih platformi koje *ElastAlert2* podržava su: *Microsoft Teams, Discord, Slack, Google Chat, Telegram, Zabbix, ServiceNow, Jira*… Također je moguće slati upozorenja preko POST metode na API te preko e-pošte. Unutar konfiguracije upozorenja se definiraju vrijednosti koje će upozorenje slati. Moguće je slati podatke samog dokumenta koji je okinuo pravilo, podatke o samom pravilu te unaprijed definirane podatke zadane u pravilu. Podaci unutar upozorenja mogu se i dinamički definirati korištenjem *Python3 Jinja2* paketa koji omogućava rudimentarnu logiku kod definiranja sadržaja. Sadržaj upozorenja razlikuje se s obzirom na platformu na koju se upozorenje šalje [49].

2.3. VirusTotal

VirusTotal je često korišten alat u kibernetičkoj sigurnosti, koji omogućava analizu datoteka i URL-ova radi otkrivanja zlonamjernog softvera i drugih prijetnji. Njegove primarne funkcije uključuju analizu zlonamjernog softvera, procjenu sigurnosti datoteka i URL-ova, obavještavanje o prijetnjama i odgovore na incidente. *VirusTotal* koristi više antivirusnih servisa kako bi nad predanim resursom odradio analizu ponašanja, provjeru u kontroliranim uvjetima te primjenu strojnog učenje za generiranje sveobuhvatnih izvješća koja pomažu u identifikaciji i ublažavanju prijetnji. Sigurnosni stručnjaci koriste ovu platformu za dobivanje uvida u nove prijetnje, analizu obrazaca napada i poboljšanje razumijevanja kibernetičkog krajolika. Lako se integrira u postojeću sigurnosnu infrastrukturu, omogućavajući automatiziranu analizu i upozorenja u stvarnom vremenu. Integracija se odvija kroz robusno implementiran API, koji kao odgovor vraća sve podatke o analizi resursa. Njegova sposobnost brzog otkrivanja i ublažavanja prijetnji posebno je posebno važna u kontekstu napada nultog dana, gdje tradicionalne metode često zakažu. Korištenjem višestrukih antivirusnih servisa i sigurnosnih alata, *VirusTotal* omogućuje sveobuhvatnu sigurnosnu procjenu te tako pokriva najveći mogući prostor vektora napada. Svaki upit te njegova analiza se spremaju u bazu podataka te se njima može u budućnosti pristupiti kao dijelu javne domene. Na ovaj način se osigurava visoka razina kolaboracije među stručnjacima te drugim alatima koji koriste *VirusTotal* kao bazu podataka[50].

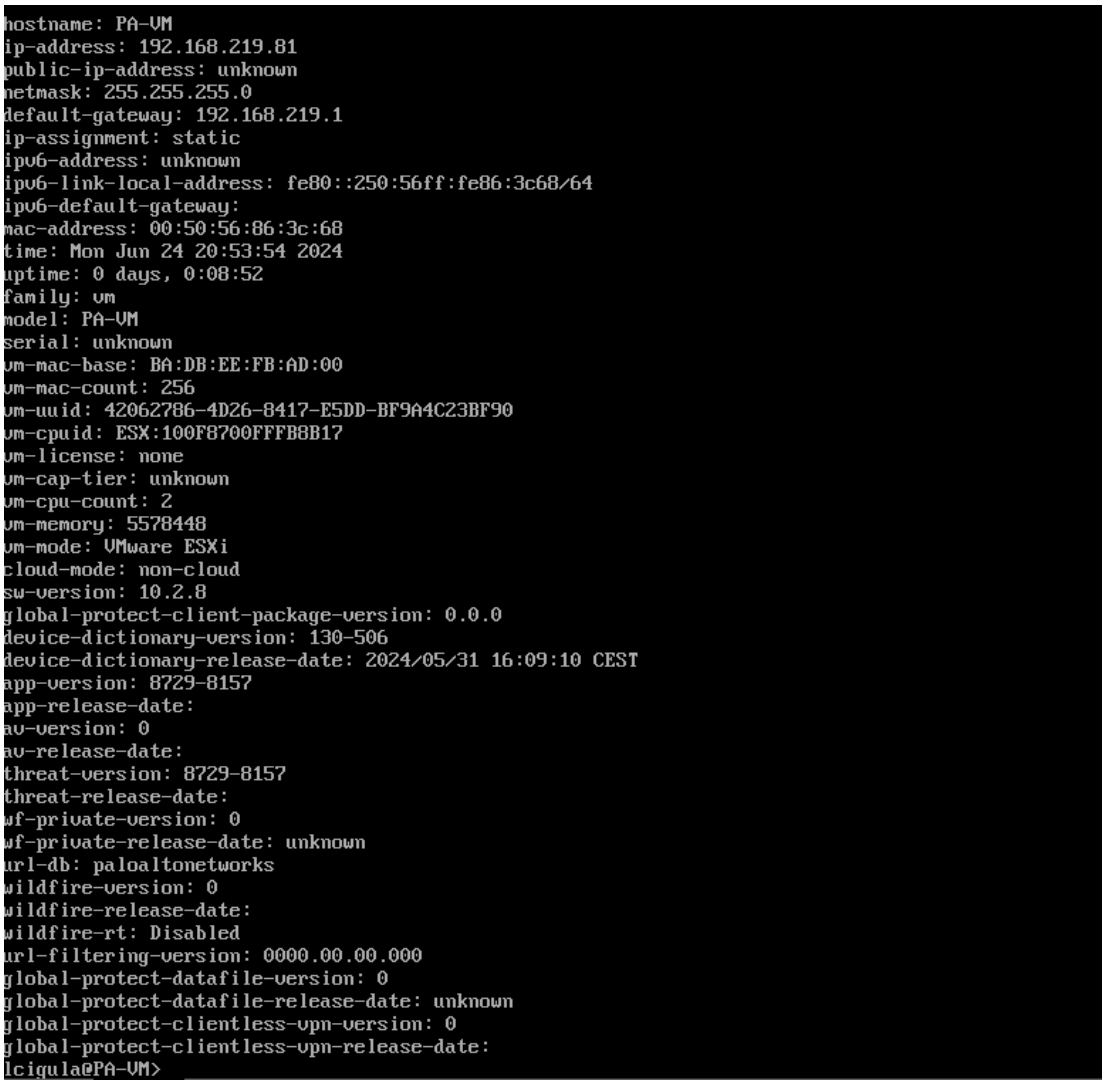
2.4. PAN-OS

PAN-OS je operativni sustav kojeg je razvila tvrtka *Palo Alto Networks*, koja se pozicionirala kao jedan od vodećih pružatelja rješenja za mrežnu sigurnost. Dizajniran je za rad na vatrozidima nove generacije, s ciljem integracije velikog broja sigurnosnih funkcionalnosti u jedinstvenu platformu. Povijest PAN-OS-a počinje osnivanjem kompanije *Palo Alto Networks* 2005., dok su prvi uređaji s PAN-OS-om izdani 2007. godine te su s njim došle značajne promjene u industriji mrežne sigurnosti. Kroz pružanje usluge napredne analize prometa te olakšane administracije vatrozida, PAN-OS je postao okosnica razvoja vatrozida nove generacije. Razvoj PAN-OS-a nastavljen je kroz kontinuirana poboljšanja i nadogradnje koje su uključivale nove funkcionalnosti poput integracije sa servisima u oblaku, napredne zaštite od prijetnji kroz integraciju s drugim pružateljima sigurnosti te

generalnih poboljšanja performansi. PAN-OS koristi *CentOS Linux* distribuciju kao temelj za svoj operacijski sustav. Posljednja verzija je *v11.2 Quasar* [51].

2.4.1. Sustav upravljanja

PAN-OS kao sustav upravljanja izlaže tri metode interakcije. Osnovni način interakcije je programska ljuska, odnosno naredbeni redak (eng. *Command Line – CLI*) prikazan na slici (Slika 2.3). Ova vrsta upravljanja namijenjena je za napredne korisnike, no u praksi se uglavnom koristi samo prilikom postavljanja uređaja ili otklanjanja većih grešaka. Može biti korisna i za automatizaciju, no kao najveći nedostatak ističe se kompleksnost upravljanja. Naredbeni redak je sada već gotovo arhaični pristup koje je zamijenilo web sučelje. Web sučelje pruža kompletan pristup svim funkcionalnostima PAN-OS-a. Web sučelje omogućava brz i jednostavan pristup upravljanja samim vatrozidom. Pruža pregledan uvid u sve konfiguracijske opcije te trenutno stanje uređaja. Manipulacija objektima te implementacija i izmjena pravila znatno su olakšane korištenjem Web sučelja. Dodatna funkcionalnost je lak pregled dnevničkih zapisa samog uređaja te konekcija koje kroz njega prolaze. Konačno, automatizacija PAN-OS sustava odvija se kroz API upite. Logika iza API upita slijedi objektni model, gdje se svaki dio PAN-OS sustava, poput sučelja, mrežnih adresa ili pravila klasificira kao objekt. Objekti se mogu kreirati te se s njima može manipulirati. Ovakav pristup omogućava širok spektar mogućnosti nad svakim od objekata, no značajno podiže kompleksnost pristupa. Kako bi se proces automatizacije olakšao, koristi se *Panos Python3* biblioteka kao alat za API interakciju s vatrozidom. Ova biblioteka omogućuje sveobuhvatno upravljanje konfiguracijom uređaja, što uključuje kreiranje, brisanje i izmjenu različitih mrežnih objekata. Dodatno, pruža sposobnosti za upravljanje sigurnosnim politikama, NAT pravilima, QoS pravilima i pravilima usmjeravanja na temelju politika. Također omogućava dohvat informacija o sustavu te svim njegovim objektima. Vodi se dvojnim principom konfiguriranja, gdje sam program čuva jedan objekt vatrozida, sa svim njegovim karakteristikama kao nasljednim objektima te se taj objekt sinkronizira sa „stvarnim“, udaljenim vatrozidom po potrebi. Ovakav pristup pruža modularan i siguran način izmjene konfiguracija jer se svaka konfiguracija po potrebi može testirati i lokalno, prije implementacije na udaljenom vatrozidu [52].



Slika 2.3 CLI sučelje na *Palo Alto* vatrozidu

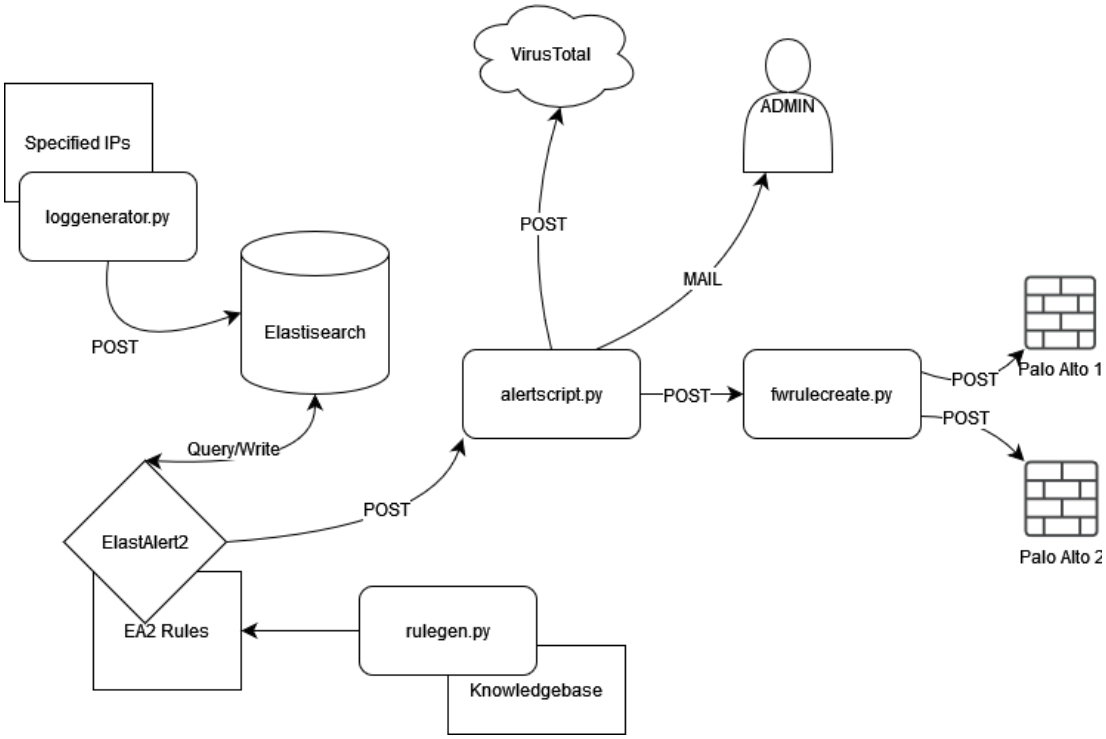
3. Opis rješenja

U ovom poglavlju je opisana arhitektura i izvedba samog simuliranog rješenja. Rješenje se tijekom izrade rada izvodilo u kombinaciji virtualnih strojeva na udaljenim mjestima.

3.1. Arhitektura rješenja

Program *log_generator.py* generira javne i privatne IP adrese te izmišljene mrežne dnevničke zapise koji se zatim putem HTTP POST metode šalju u *Elasticsearch* grozd. Tamo se zapisi spremaju te se istima može pristupiti preko API-a izloženog na mrežnim vratima 9200 ili pristupom *Kibani* na mrežnim vratima 5601. Zapisi se čuvaju u indeksu *network_logs* koji je pretvoren u fragmente čija se redundancija osigurava na istom čvoru. Podatke sačinjava vremenski žig, količina podataka (u bajtovima), IP adrese te broj paketa. Podaci se mogu sortirati s obzirom na vrijeme generiranja. Na grozd je spojen *ElastAlert2* sustav koji s obzirom na definirane politike aktivira pravila te šalje upozorenja programu za obradu i procesiranje upozorenja. *ElastAlert2* pravila se mogu pisati ručno u direktoriju *./rules* ili pak kroz definiranje IP adresa za koje korisnik želi da se pravila generiraju u datoteci *specified_ips.txt*. *ElastAlert2* provodi upite nad skupom podataka u unaprijed definiranom vremenskom razdoblju. Ispituje se isključivo indeks *network_logs. ElastAlert2* Također definira svoje indekse unutar kojih zapisuje podatke o svojim radnjama. Unutar tih indeksa su zapisana vremena aktiviranja pravila te vremena i podaci o upitima koji su izvršeni nad skupom podataka. Pristup ovim indeksima je također omogućen kroz sučelje *Kibana*. Kada se pravilo aktivira, sukladno konfiguraciji definirano je slanje HTTP POST zahtjeva prema servisu kojim upravlja program za generiranje upozorenja. Upravo ovaj servis se brine za izlaganje krajnje točke koja prima JSON s podacima o IP izvorištu i odredištu te smjeru komunikacije. Sukladno ovim podacima se ispituje vanjska IP adresa na servisu *VirusTotal* te se o istoj skupljaju relevantni podaci. Ovi podaci se šalju na proizvoljnu adresu e-pošte kako bi se obavijestio administrator sustava. Također, IP adrese se šalju novom HTTP POST metodom na servis za slanje zahtjeva na vatrozide kako bi se generiralo pravilo za obustavu komunikacije između navedenih adresa. Kod svakog slanja se unaprijed provjerava postoji li isto pravilo otprije. Također se provjerava postoje li objekti sučelja, zadanih IP adresa te zona sučelja. Ako ne postoje, upisuju se u bazu objekata samog

vatrozida te se njima može manipulirati i ručno kroz web sučelje vatrozida. Dijagram arhitekture prikazan je na slici (Slika 3.1).



Slika 3.1 Dijagram arhitekture rješenja

3.2. SIEM servisi

Servisi korišteni u opsegu ovog rada za primarnu funkciju imaju simulaciju SIEM sustava. U tu svrhu se koristi kao baza u kojoj se čuvaju dnevnički zapisi generirani programom. Kao sučelje za pristup i vizualizaciju podataka koristi se *Kibana.* Konačno, kao sustav za generiranje upozorenja koristi se *ElastAlert2.* Svi servisi su kontejnerizirani korištenjem tehnologije *Docker* u svrhu lakšeg održavanja i manipulacije.

3.2.1. Provizioniranje

Provizioniranje svih potrebnih servisa se odrađuje kroz *Docker Compose* pomoćni alat. Konfiguracija se čuva u *docker-compose.yml* datoteci u kojoj se definira osnovno ponašanje servisa, njihova povezanost sa sustavom domaćina te njihove mrežne karakteristike. Konfiguracija koristi *Docker Compose* verziju 3.8 te su definirane tri ključne usluge:

Elasticsearch, *Kibana* i *ElastAlert2*, koje djeluju unutar zajedničke Docker mreže nazvane `elastic`. *Elasticsearch* koristi službenu *Docker* sliku verzije 8.13.0, s imenom kontejnera `elasticsearch`. Postavljen je u način rada s jednim čvorom, s isključenim sigurnosnim karakteristikama. Ovakav pristup je odabran kako bi se smanjio utrošak resursa te olakšala implementacija i integracija s ostalim dijelovima složaja. Memorijski limiti su definirani s `ES_JAVA_OPTS` opcijom. Definiranje memorijskih limita je izrazito važno kod servisa, s obzirom na to da upravo oni definiraju brzinu i opseg samog sustava. U okviru ovog rada je 512MB više nego dovoljna količina radne memorije za kratkoročno čuvanje i obradu podataka. Podaci se pohranjuju lokalno u kontejner. Kontejner izlaže prema vanjskoj mreži mrežna vrata 9200 kako bi se moglo manipulirati grozdom s API upitima upućenim sa sustava domaćina. *Kibana*, koristeći sliku verzije 8.13.0, ovisi o usluzi te omogućuava vizualizaciju podataka. Koristi se varijabla `ELASTICSEARCH_HOSTS` za povezivanje s uslugom unutar *Docker* mreže. Izložena su mrežna vrata 5601 te je usluga povezana s mrežnim sučeljem i IP adresom domaćina kako bi se sučelju moglo pristupiti i s udaljene mrežne točke. *ElastAlert2* koristi sliku *jertel/elastalert2*, automatski se ponovno pokreće u slučaju greške i koristi lokalne konfiguracijske datoteke. Ova usluga kontinuirano nadzire podatke iz *Elasticsearch*a i generira upozorenja prema definiranim pravilima. Kako su konfiguracijske datoteke direktno povezane s datotečnim sustavom domaćina, ovakav pristup omogućava mijenjanje konfiguracija tijekom rada samog kontejnera. S obzirom na dinamičkog mijenjanje i automatsko uvođenje pravila, ovo funkcionalnost je nužna jer bi se u suprotnom sustav morao gasiti kod svake promjene.

3.3. Implementacija programske logike

Dodatna potrebna logika koja implementira odnose između dijelova sustava je izvedena korištenjem *Pyton3* programskog jezika u obliku programskih skripti.

3.3.1. Generiranje dnevničkih zapisa

Python3 program log_*generator*.py stvara nasumične IP adrese ili koristi unaprijed definirane IP adrese iz tekstualne datoteke kako bi generirala dnevničke zapise o mreži te iste slala u grozd. Program definira klasu `LogGenerator`, koja se inicijalizira s putanjom do datoteke koja sadrži javne IP adrese i definira lokalne IP mreže koje predstavljaju privatne IP prostore. Metoda `generate_log` nasumično odabire jednu lokalnu mrežu i generira

nasumičnu lokalnu IP adresu unutar te mreže, dok se javna IP adresa generira metodom `_generate_public_ip`, koja osigurava da IP adresa nije dio lokalnih mreža. IP adrese se potom nasumično dodjeljuju izvoru i odredištu te se određuje smjer prometa kao dolazeći ili odlazeći. Broj paketa i veličina u bajtovima također se nasumično generiraju, dok se trenutni vremenski žig dobiva u ISO formatu kao trenutno vrijeme. Kreirani log zapis se pohranjuje u JSON objektu. Alternativno, metoda `generate_log_from_file` učitava javne IP adrese iz zadane datoteke, odabire nasumičnu javnu IP adresu i lokalnu IP adresu te kreira log zapis na sličan način kao i metoda `generate_log`. Funkcija `send_log_to_elasticsearch` postavlja URL servisa i zaglavlja HTTP zahtjeva te šalje log zapis koristeći HTTP POST zahtjev, vraćajući statusni kod i odgovor servisa. Asinkrona funkcija `periodic_log_generation` omogućuje periodičko generiranje logova i njihovo slanje na u redovitim intervalima, koristeći metodu `generate_log_from_file` za generiranje logova. Na kraju, program inicijalizira objekt `LogGenerator` s putanjom do datoteke s javnim IP adresama te pokreće asinkronu funkciju `periodic_log_generation`. Ovaj program osigurava automatizirano generiranje i slanje mrežnih zapisa, koji su unutar ovog rješenja baza na kojoj se ostatak tehničkog dijela rada temelji.

3.3.2. Obrada upozorenja

Za obradu upozorenja koristi se *Flask* programska biblioteka kroz koju se implementira logika primanja HTTP POST zahtjeva. Također implementira obradu JSON podataka te korištenje vanjske API usluge *VirusTotal* za provjeru IP adresa. Unutar *Flask* aplikacije definirana je ruta */alert* koja prihvaća POST zahtjeve koje prima od sustava *ElastAlert2*. Kada se primi zahtjev, funkcija `receive_post` dohvaća neobrađenes podatke iz zahtjeva kao tekstualni niz (`raw_data`) i ispisuje ih radi praćenja. Dobiveni JSON podaci se obrađuju te ako obrada uspije, izdvajaju se vrijednosti za ključ `source_ip` iz JSON objekta. Ako je `source_ip` prisutan, funkcija `query_virustotal` se poziva s IP adresom kao argumentom. Rezultat poziva funkcije `query_virustotal` ispisuje se i vraća kao odgovor HTTP zahtjeva. Sukladno poslanoj IP adresi kreira se URL za *VirusTotal* API koristeći formatirani niz s IP adresom. Zaglavlja HTTP zahtjeva definirana su s API ključem za autorizaciju pristupa *VirusTotal* API-u. Izvodi se GET zahtjev na *VirusTotal* API koristeći `requests.get`. Ako je HTTP statusni kod odgovora 200, vraća se JSON odgovor *VirusTotal* API-a zajedno s HTTP statusnim kodom 200. Također, poziva se funkcija `send_email_gmail` koja koristeći

Googleov SMTP poslužitelj šalje e-poštu prema proizvoljnoj adresi. Kao sadržaj, pošta šalje analizu dobivenu s *VirusTotala*. Ako poziv ne uspije, vraća se odgovarajuća poruka o pogrešci i HTTP statusni kod odgovora.

3.3.3. Generiranje *ElastAlert2* pravila

Ova *Python3* skripta omogućava validaciju IP adresa i automatsko generiranje pravila za alat *ElastAlert2* u YAML formatu, čime se olakšava proces detekcije i obavještavanja o potencijalno zlonamjernom mrežnom prometu. Funkcija `create_elastalert_rule` kreira YAML datoteku koja sadrži pravilo za ElastAlert na temelju zadane IP adrese. Ime datoteke uključuje IP adresu, a sadržaj YAML datoteke definiran je kao *Python3* rječnik koji sadrži naziv pravila, tip, indeks u *ElasticSearchu*, filter koji traži zapise gdje je polje `src` jednak zadanoj IP adresi, vrijeme između ponovljenih upozorenja te postavke za HTTP POST zahtjev koji se šalje kada se pravilo aktivira. Datoteka se sprema u zadani direktorij koristeći modul `yaml` za pretvaranje rječnika u YAML format. Funkcija `process_ip_addresses` otvara datoteku koja sadrži IP adrese i čita ih liniju po liniju. Za svaku IP adresu, funkcija poziva `is_valid_ip` metodu da provjeri valjanost IP adrese. Ako je IP adresa valjana, kreira se pravilo pomoću `create_elastalert_rule`, u suprotnom, ispisuje se poruka o nevaljanoj IP adresi. Program omogućuje automatiziranu validaciju IP adresa i kreiranje pravila za alat *ElastAlert2*, čime se značajno unapređuje proces detekcije i odgovora na mrežne prijetnje.

3.3.4. Kreacija pravila na vatrozidu

Kreacija pravila na vatrozidu implementirana je kroz *Python3* program koristeći programsku biblioteku *Flask*. Korištenjem *Flask* biblioteke implementirano je API sučelje koje prima podatke potrebne za kreiranje sigurnosnih pravila na *Palo Alto* vatrozidu putem HTTP POST zahtjeva. Funkcija `receive_post` prima JSON podatke koji sadrže izvorne i odredišne IP adrese te poziva funkciju `create_fw_rule`, koja inicijalizira vezu s vatrozidom koristeći zadane detalje poput IP adrese i API ključa. Unutar funkcije `create_fw_rule` provjerava se postojanje potrebnih mrežnih zona (*untrust* i *trust*), koje, ako ne postoje, kreiraju se. Sljedeći korak uključuje provjeru i eventualno kreiranje fizičkih mrežnih sučelja (*ethernet1/1* i *ethernet1/2*), te pripadajućih podsučelja OSI sloja 3 s VLAN oznakama. Također se kreiraju objekti adresa za izvorne i odredišne IP adrese ukoliko već ne postoje. Na kraju, kreira se sigurnosno pravilo koje blokira promet između specificiranih IP adresa, ukoliko takvo

pravilo već ne postoji. Logika kreacije pravila koristi `pan-os-python` biblioteku kao sloj apstrakcije nad API metodama samog vatrozida.. Objekti biblioteke čuvaju stanje konfiguracije lokalno te se po potrebi sinkroniziraju s udaljenim vatrozidom. U slučaju greške lokalna konfiguracija se neće primijeniti na udaljeni vatrozid. Ovakav pristup osigurava ispravnost te idempotentnost prilikom kreacije pravila.

Zaključak

Rezultati ovog rada pokazuju da automatizacija upravljanja pravilima vatrozida donosi značajne prednosti u kontekstu kibernetičke sigurnosti. Integracijom raspodijeljenog sustava obrade dnevničkih zapisa omogućeno je brzo i precizno ažuriranje pravila vatrozida. Smanjeno je vrijeme reakcije na prijetnje te je minimizirana mogućnost ljudskih pogrešaka. Buduća istraživanja trebala bi se fokusirati na testiranje rješenja u stvarnim produkcijskim okruženjima kako bi se dodatno potvrdila njegova učinkovitost i skalabilnost. Posebice bi važan bio test na velikom volumenu podataka te dodavanje dodatnih vatrozida. Ovakav test bi testirao skalabilnost samog sustava te mogućnost brze obrade velike količine podataka. Predlaže se testiranje rješenja uz korištenje produkcijskog SIEM sustava s dodatnim umreženim vatrozidima, po mogućnosti različitih proizvođača.

Literatura

[1] *Cybersecurity Statistics*, 28.04.2022., https://www.forbes.com/advisor/education/it-and-tech/cybersecurity-statistics/, Pristupljeno: 26.04.2024.

[2] Fortinet, *What is Elasticsearcha Firewall?*, https://www.fortinet.com/resources/cyberglossary/firewall, Pristupljeno: 26.04.2024.

[3] Hodeghatta, R., Nayak, U., *Firewalls, The InfoSec Handbook,* 01.1.2014., https://link.springer.com/chapter/10.1007/978-1-4302-6383-8_10

[4] Nmap.org, *Nmap Overview and Demonstratio*n, https://nmap.org/book/nmap-overview-and-demos.html, Pristupljeno: 26.04.2024.

[5] Palo Alto Networks, *The History of Firewalls|Who Invented the Firewall?,* https://www.paloaltonetworks.com/cyberpedia/history-of-firewalls, Pristupljeno: 28.04.2024.

[6] Fortinet, *Stateful Firewall,* https://www.fortinet.com/resources/cyberglossary/stateful-firewall, Pristupljeno: 28.04.2024.

[7] Liang, J., Kim, Y., *Evoultion of Firewalls: Toward Securer Network Using Next Generation Firewall,* IEEE, 04.03.2022., https://ieeexplore.ieee.org/abstract/document/9720435

[8] Ahmadi, S, *Next Generation AI-Based Firewalls: A Comparative Study,* International Journal of Comupter, 2023., https://hal.science/hal-04456265v1/file/Next_Generation_AI-Based_Firewalls_%20A_Comparative_Study.pdf

[9] Cloudflare, *What is Elasticsearcha next-generation firewall (NGFW)?,* https://www.cloudflare.com/learning/security/what-is-next-generation-firewall-ngfw/, Pristupljeno:10.05.2024.

[10] Palo Alto Networks, *What is Elasticsearcha Stateful Firewall? | Stateful Inspection Firewalls Explained,* https://www.paloaltonetworks.sg/cyberpedia/what-is-a-stateful-firewall, Pristupljeno:10.05.2024.

[11] IR, *Deep Packet Inspection (DPI): How it works and why it's important,* https://www.ir.com/guides/deep-packet-inspection, Pristupljeno:10.05.2024.

[12] Fortinet, *What is Deep Packet Inspection (DPI)?,* https://www.fortinet.com/de/resources/cyberglossary/dpi-deep-packet-inspection, Pristupljeno:10.05.2024.

[13] ManageEngine, *What is deep packet inspection?,* https://www.manageengine.com/products/netflow/what-is-deep-packet-inspection.html, Pristupljeno:10.05.2024.

[14] Check Point, *What is VPN (Virtual Private Network)?,* https://www.checkpoint.com/cyber-hub/network-security/what-is-vpn/ , Pristupljeno:15.05.2024.

[15] Wikipedia, *Virtual Private Network*, https://en.wikipedia.org/wiki/Virtual_private_network, Pristupljeno:15.05.2024.

[16] Wadhwa, S., Pal, K., *Providing Security in VPN by using Tunneling and Firewall,* https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=5faf53a501a66bcf0d7a18d347673c931f4bf0b8

[17] Palo Alto Networks, *What Are the Different Types of VPN Protocols?*, https://www.paloaltonetworks.com/cyberpedia/types-of-vpn-protocols, Pristupljeno:15.05.2024.

[18] Palo Alto Networks, *What is SSTP (Secure Socket Tunneling Protocol)?, https://www.paloaltonetworks.com/cyberpedia/what-is-sstp,* Pristupljeno:15.05.2024.

[19] Cybernews, *What is WireGuard?, https://cybernews.com/what-is-vpn/wireguard-protocol/ ,* Pristupljeno:15.05.2024.

[20] DataProt, *What is OpenVPN and How Does it Work?,* 14.07.2023., *https://dataprot.net/guides/what-is-openvpn/*, Pristupljeno:15.05.2024.

[21] Palo Alto Networks, *What is Quality of Service?*, https://www.paloaltonetworks.com/cyberpedia/what-is-quality-of-service-qos, Pristupljeno:16.05.2024.

[22] Fortinet, *What is Quality of Service (QoS) in Networking?*, https://www.fortinet.com/resources/cyberglossary/qos-quality-of-service, Pristupljeno:16.05.2024.

[23] F5, *What is SSL Decryption*?, https://www.f5.com/glossary/ssl-decryption, Pristupljeno:16.05.2024.

[24] John Arena, Palo Alto Network, *Should You Have SSL Decryption Enabled?*, *https://live.paloaltonetworks.com/t5/pancast-episodes/pancast-episode-9-should-you-have-ssl-decryption-enabled/ta-p/526755*, Pristupljeno:16.05.2024.

[25] Palo Alto Networks, *PAN-OS Documentation: Decryption Concepts*, https://docs.paloaltonetworks.com/pan-os/9-1/pan-os-admin/decryption/decryption-concepts, Pristupljeno:16.05.2024.

[26] Fortinet, *Network topology best practices*, https://docs.fortinet.com/document/fortideceptor/4.0.0/best-practices/557405/network-topology-best-practices

[27] Fortinet, *DMZ Networks*, https://www.fortinet.com/resources/cyberglossary/what-is-dmz, Pristupljeno:26.05.2024.

[28] Beyond Trust, *Systems Hardening*, https://www.beyondtrust.com/resources/glossary/systems-hardening, Pristupljeno:26.05.2024.

[29] Firewall.cx, *Next-Gen Firewalls & Topologies. Designing & Building DMZs. Concepts, Best Practices & Tips*, https://www.firewall.cx/networking/network-fundamentals/firewall-topologies-dmz-zone.html, Pristupljeno:26.05.2024.

[30] Ostec, *Understanding the main firewall topologies*, https://ostec.blog/en/perimeter/understanding-the-main-firewall-topologies/, Pristupljeno:26.05.2024.

[31] Ioannidis, S., A.,D., Keromytis, Bellovin, M., S., Smith., M., J., *Implementing Elasticsearcha Distributed Firewall*, https://www1.cs.columbia.edu/~smb/papers/ccs-df.pdf

[32] Barracuda, *Distributed Firewall*, https://www.barracuda.com/support/glossary/distributed-firewall, Pristupljeno:26.05.2024.

[33] Bellovin, M., S., *Distributed Firewalls*. https://www.cs.columbia.edu/~smb/papers/distfw.pdf, Pristupljeno:26.05.2024.

[34] Elastic, *What is Elasticsearch?* https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html, , Pristupljeno:04.06.2024.

[35] Elastic, *Designing for resilience,* https://www.elastic.co/guide/en/elasticsearch/reference/current/high-availability-cluster-design.html, Pristupljeno:04.06.2024.

[36] Elastic, *Voting configurations,* https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-discovery-voting.html, Pristupljeno:04.06.2024.

[37] Elastic, *Node,* https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-discovery-voting.html, Pristupljeno:04.06.2024.

[38] Jake Landis, *Implementing a Hot-Warm-Cold Architecture with Index Lifecycle Management*, 10.04.2019, https://www.elastic.co/blog/implementing-hot-warm-cold-in-elasticsearch-with-index-lifecycle-management., Pristupljeno:04.06.2024.

[39] BlueXP, *Elasticsearch Architecture: 7 Key Components,* 10.05.2021, https://bluexp.netapp.com/blog/cvo-blg-elasticsearch-architecture-7-key-components, Pristupljeno:04.06.2024.

[40] Morten Ingebrigtsen, *Indexing for Beginners, Part 2,* 08.10.2023., https://www.elastic.co/blog/found-indexing-for-beginners-part2, Pristupljeno:04.06.2024.

[41] Bo Andresen, *Understanding the Inverted Index in Elasticsearch,* 05.05.2018., https://codingexplained.com/coding/elasticsearch/understanding-the-inverted-index-in-elasticsearch, Pristupljeno:04.06.2024.

[42] Eleanor Bennett, *What is Kibana?*, 08.02.2024, https://logit.io/blog/post/what-is-kibana/, Pristupljeno:04.06.2024.

[43] Tech Tim, *Elastic Search and Open Search: a brief history of the license war,* 08.03.2023., https://medium.com/@TechTim42/elastic-search-and-open-search-a-brief-history-of-the-license-war-8f474743e2ff, Pristupljeno:04.06.2024.

[44] George Kobar, Ugo Sangiorgi, *Elasticsearch vs. OpenSearch: Unraveling the performance gap*, 08.08.2023., https://www.elastic.co/blog/elasticsearch-opensearch-performance-gap, Pristupljeno:04.06.2024.

[45] ElastAlert2 Documentation, *Overview,* https://elastalert2.readthedocs.io/en/latest/elastalert.html, Pristupljeno:07.06.2024.

[46] ElastAlert2 Documentation, *Getting Started,* https://elastalert2.readthedocs.io/en/latest/running_elastalert.html, Pristupljeno:07.06.2024.

[47] ElastAlert2 Documentation, *Operational Review,* *https://elastalert2.readthedocs.io/en/latest/running_elastalert.html#operational-review*, Pristupljeno:07.06.2024.

[48] ElastAlert2 Documentation, *Rule Types,* https://elastalert2.readthedocs.io/en/latest/ruletypes.html#rule-types, Pristupljeno:07.06.2024.

[49] ElastAlert2 Documentation, *Alerts*, https://elastalert2.readthedocs.io/en/latest/alerts.html, Pristupljeno:07.06.2024.

[50] Infosec-Jobs, *VirustTotal Explained*, 06.12.2023., https://infosec-jobs.com/insights/virustotal-explained/, Pristupljeno:07.06.2024.

[51] PaloAlto TechDocs, *PAN-OS,* https://docs.paloaltonetworks.com/pan-os, Pristupljeno:12.06.2024.

[52] PaloAlto TechDocs, *Firewall Administration*, https://docs.paloaltonetworks.com/pan-os/10-2/pan-os-admin/firewall-administration, Pristupljeno:12.06.2024.

Sažetak

Rad je fokusiran na problematiku automatizacije kreacije pravila na vatrozidima u sustavima zaštite kibernetičkog prostora. Kako ručno upravljanje pravilima vatrozida postaje neefikasno te sklono pogreškama, u radu je predstavljeno rješenje automatizacije pravila. Korištene su tehnologije za skupljanje i obradu dnevničkih zapisa poput *Elasticsearcha* i *ElastAlerta2* te je kroz *Python3* programski jezik implementiran sustav za kreaciju pravila na *Palo Alto* vatrozidima. Ovakva integracija omogućava brzu reakciju na prijetnje te značajno smanjuje potrebu za ručnim intervencijama, što kao rezultat ima povećanje kvalitete i brzine sigurnosnog odziva čitavog sustava.

Summary

The paper focuses on the issue of automating the creation of firewall rules in cyber defense systems. As manual management of firewall rules becomes inefficient and prone to errors, the paper presents a solution for rule automation. Technologies for collecting and processing log records, such as *Elasticsearch* and *ElastAlert2*, were used. A system for creating rules on Palo Alto firewalls was implemented through the Python3 programming language. This integration enables a swift response to threats and significantly reduces the need for manual interventions, resulting in increased quality and speed of the overall system's security response.

Privitak

U ovom poglavlju su navedene upute za instalaciju i pokretanje.

Struktura direktorija sa svim potrebnim alatima:

```
lc_dipl_solution
|
|
+-------eslab
|        |
|        |
|        +------rules
|        |       |
|        |       +-----rule_1.yaml
|        |       |
|        |       +-----rule_2.yaml ...
|        |
|        +------docker-compose.yaml
|        |
|        +------elastalert.yaml
|
|
+-------scripts
|        |
|        |
|        +------data
|        |       +-----knowledgebase.txt
|        |       |
|        |       +-----specified_ips.txt
|        |
|        +------.env
|        |
|        +------loggenerator.py
|        |
|        +------alertscript.py
|        |
|        +------fwrulecreate.py
|        |
|        +------rulegen.py
|
+-------requirements.txt
```

Rješenje je testirano na *Debian 12* i *RedHat 9* distribucijama *Linux* operativnog sustava. U ostatku teksta se podrazumijeva da je korisnik pozicioniran u direktorij 'lc_dipl_solution'. Potrebna programska podrška za pokretanje je:

- *Python3* programski jezik i *Pip3* kao sustav za upravljanje paketima
- *Docker* i *Docker Compose* sustavi za kontejnerizaciju

- Generiran API ključ za *VirusTotal*
- Generirani API ključevi za *PAN-OS* sustave

Svi potrebni Python3 paketi mogu se instalirati pomoću naredbe:

```
pip3 install -r requirements.txt
```

Kako bi se rješenje pokrenulo, potrebno je pokrenuti *Elasticsearch* labos. Prvo u eslab/elastalert.yaml treba izmijeniti 'es_host' u IP adresu na kojoj će se pokretati *Elasticsearch*. Labos se pokreće izvršavanjem komande:

```
docker compose -f eslab/docker-compose.yaml up -d
```

Na ovaj način će se pokrenuti *Elasticsearch, Kibana i ElastAlert2* sustavi kao kontejneri u pozadini. Kako bi pristupili web sučelju *Kibane*, potrebno je otići na web sjedište http://HOST_IP:5601, pri čemu je HOST_IP IP adresa poslužitelja na kojem su sustavi pokrenuti. Labos se može ugasiti s naredbom:

```
docker compose -f eslab/docker-compose.yaml down
```

Sve programske skripte i potrebni podaci nalaze se u 'scripts/' direktoriju. Sve skripte se pokreću s komandom:

```
python3 ime_skripte.py'
```

Svaka skripta se tretira kao poseban proces te je za istu potrebno odgovarajuće okruženje.

'.env' datoteka posjeduje sve varijable okruženja u kojima se nalaze ključni podaci za izvedbu rješenja. Istu treba kreirati prije pokretanja skripti! Sadrži sljedeće varijable:

- ES_HOST_AND_PORT="IP:PORT" (npr. " 192.168.1.2:9200")
    - Sadrži informacije o IP adresi i mrežnim vratima za pristup *Elasticsearchu*
- EA2_ALERT_ENDPOINT_HOST_AND_PORT="IP:PORT", (npr. " 192.168.1.2:6000")
    - IP adresa i mrežna vrata skripte za obradu upozorenja
- RULE_CREATE_ENDPOINT_HOST_AND_PORT="IP:PORT", (npr. " 192.168.1.2:6001")

    - IP adresa i mrežna vrata skripte za kreiranje pravila na vatrozidu
- MAIL_USER=“user@mail.com“
    - Adresa e-pošte s koje će se slati upozorenje
- MAIL_PASS=“pass123“
    - Lozinka e-pošte s koje se šalje upozorenje
- ADMIN_MAIL=“admin@mail.com“
    - Adresa e-pošte na koju se šalje upozorenje
- VIRUSTOTAL_API_KEY=“api_key“
    - API ključ za pristup VirusTotal API-u
- FW_PA_IP_LIST=“FW_IP1, FW_IP2“, (npr. “192.168.1.10, 192.168.1.11“)
    - IP adrese PAN vatrozida, odvojene zarezom i razmakom (“, “)
- FW_PA_KEY_LIST=“FW_KEY1, FW_KEY2“
    - API ključevi PAN vatrozida, odvojeni zarezom i razmakom (“, “)

`loggenerator.py` skripta generira zapise te ih šalje u *Elasticsearch*. S vjerojatnošću od 50% se generira ili potpuno nasumični zapis ili zapis koji sadrži nasumičnu adresu iz 'specified_ips.txt'. Adrese u ovoj datoteci su zapisane tako da je svaka u novom retku. Na standardni izlaz se ispisuje svaki generirani zapis.

`alertscript.py` skripta pokreće API krajnju točku na kojoj očekuje upozorenje s *ElastAlert2* sustava. Zatim šalje podatke o adresama prema skripti za kreiranje pravila. Primitak upozorenja te njegovo slanje se ispisuju na standardni izlaz. Također, sukladno upozorenju prima podatke o IP adresi s *VirusTotal* API-a te iste šalje na adresu specificiranu u '.env' datoteci.

`fwrulecreate.py` prima podatke za kreiranje pravila te pravilo i kreira na svim uređajima specificiranim u '.env' datoteci. Sva pravila se izvršavaju na uređajima te tamo ostaju dok ih se ručno ne obriše.

`rulegen.py` skripta služi kao pomoćna skripta za kreiranje *ElastAlert2* pravila. Kreirat će se pravila za sve adrese specificirane u knowledgebase.txt datoteci. IP adrese u knowledgebase.txt trebaju biti navedene svaka u svojem redu.