SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 6820

Generiranje korisničkog prometa na kibernetičkim poligonima

Ivan Ivanković

Zagreb, lipanj 2020.

Sažetak

Povećavanjem kompleksnosti infrastrukture, povećava se i područje prijetnje koje može biti iskorišteno za napad. Kako bi se doskočilo tom problemu, tvrtke ulažu u edukaciju zaposlenika s ciljem da dobiju znanje potrebno kako bi se obranili od napada. Osmišljena su virtualna okruženja koja služe praktičnom uvježbavanju kibernetičke sigurnosti i ta okruženja se nazivaju kibernetičkim poligonima. Kibernetički poligoni trebaju što više ličiti stvarnim sustavima, jer nije poželjno vježbati kibernetičku sigurnost na stvarnim komponentama. Kako bi mreža kibernetičkog poligona što više ličila stvarnoj mreži potrebno je da tom mrežom kolaju podaci koji nalikuju podacima koje generira korisnik. Komponenta kibernetičkog poligona koja to ostvaruje se naziva generatorom prometa. U radu je cilj da se objasne osnovne karakteristike kibernetičkih poligona, te zatim potraži rješenje za generator prometa. Opisani su razni alati, njihove prednosti i mane i na kraju je odabran jedan alat. Kreirana je virtualna okolina u kojoj je taj alat testiran i snimljen je generiran promet. Nad generiranim prometom je odrađena analiza podataka, koji protokoli su se koristili prilikom generiranja podataka, kojih su veličina paketi generirani te kako izgledaju grafovi generiranih paketa po sekundi. Lokalne mreže sa stvarnim korisnicima imaju fraktalna svojstva, te je zato proučeno svojstvo usko vezano s fraktalima, a to je samo-sličnost. Zatim je dokazana samo-sličnost generiranih podataka i uspoređena s podacima koji su dobiveni u drugom radu koji je računao samo-sličnost lokalne mreže sa stvarnim korisnicima. Time je dokazano da su generirani podaci slični kao podaci koje su proizveli stvarni korisnici i da bi se ovakav tip generatora prometa uistinu mogao koristiti u kibernetičkom poligonu.

Ključne riječi: kibernetički poligon, generiranje korisničkog prometa, kibernetička sigurnost

Abstract

By increasing complexity of infrastructure, field of attack is also increasing. To mitigate that problem, company's are investing in education of their employees so that they could defend infrastructure against attacks. Cyber ranges are virtual environments used to train and teach people about cyber security. Cyber ranges should simulate real infrastructure so that real infrastructure is not at risk while doing exercise. Real network has user data going through it and for that reason cyber range should also simulate user traffic in virtual network. Part of cyber range that generates user like data is called traffic generator. Goal of this thesis is to explain main characteristics of cyber range and propose solution for traffic generator. Many tools are used, explained how they work, but also what are their flaws. After useful tool is found, virtual environment is created, tool is tested and traffic is recorded for further analysis. Local networks have fractal properties and fractals are self-similar. To prove that generated traffic has similar properties to traffic generated by real users, self-similarity is proved. These results are compared to results from other research and results prove that generated traffic has similar properties to traffic generated by real users.

Key words: cyber range, generating user traffic, cyber security

Sadržaj

1. Uvod

Ubrzanim razvojem tehnologije povećava se i njena kompleksnost. Kompleksnost alata koji se koriste u sustavima smanjuje preglednost i mogućnost da administratori sustava imaju uvid u radnje koje se izvršavaju unutar alata. Smanjenjem preglednosti i korištenjem alata koji nisu otvorenog koda korisniku alata se oduzima mogućnost uviđanja sigurnosnih propusta koje možda alat posjeduje. Zbog smanjene preglednosti korisnici alata često ne izvršavaju penetracijske testove nad alatima koje koriste jer pretpostavljaju da su sigurni. Nažalost, postoje ljudi koji su spremni napraviti detaljno penetracijsko testiranje alata i pronalaskom tih mana, zlouporabiti sigurnosne propuste u sustavima kako bi ostvarili korist. Ako dođe do napada na sustav, administratori sustava bi trebali znati kako se obraniti od napada, no to često nije slučaj.

Sljedeće statistike ukazuju na razmjer problema:

- tvrtkama u prosjeku treba 197 dana da bi otkrile da se dogodio sigurnosni propust i 69 dana da bi ga uklonile [1],
- broj sigurnosnih ispada je narastao za 11% između 2018. i 2019. godine, a 67% između 2014. i 2019. godine [2],
- 30% sigurnosnih ispada 2020. godine je uzrokovano zbog unutrašnjih faktora [3],
- 43% napada 2020. godine je izvršeno nad internetskim aplikacijama [3].

Radi porasta napada i sigurnosnih propusta, pojavila se potražnja za rješenjem na tog problema. Propuste nije moguće u potpunosti spriječiti, ali dodatnim educiranjem je moguće ukazati na potencijalne sigurnosne propuste koje su administratori sustava previdjeli. Brojne tvrtke investiraju u ovu vrstu osposobljavanja zaposlenika s ciljem da, ako dođe do napada, zaposlenici posjeduju potrebno znanje i vještine kako bi adekvatno reagirali na napad. Učenje uz vježbu i teorijsku podlogu obično daje bolje rezultate nego učenje same teorije i zato su osmišljeni kibernetički poligoni (engl. cyber ranges). Kibernetički poligoni su virtualna okruženja na kojima se mogu uvježbavati mnoge vrste napada, ali isto tako kako se obraniti od napada. Oni nude sigurno okruženje koje ne dovodi u pitanje sigurnost postojeće infrastrukture prilikom uvježbavanja. Pokretanjem virtualnog okruženja kibernetičkog poligona teku samo podaci koji su generirani od raznih alata unutar virtualnog okruženja. Problem pri tome je da se promet generiran od alata unutar virtualnog okruženja jednostavno filtrira. Nakon što je promet alata virtualnog okruženja filtriran nije teško vidjeti aktivnosti korisnika poligona. Takav scenarij znatno smanjuje kvalitetu vježbe, jer tijekom odvijavanja vježbe na kibernetičkom poligonu suprotstavljeni timovi se mogu međusobno prisluškivati. U mrežama na kojima se nalaze stvarni korisnici koji pretražuju internetom postoji šum koji otežava filtriranje podataka, a time i prisluškivanje.

U želji da poligon nalikuje stvarnoj infrastrukturi potrebno je ponuditi rješenje tog problema. Kako bi se riješio taj problem, svaki kibernetički poligon treba imati generator korisničkog prometa koji će simulirati promet koji teče infrastrukturom kao i na stvarnoj infrastrukturi.

Cilj rada je izraditi adekvatni generator korisničkog prometa te ispitati njegovu učinkovitost. Prije rada na generatoru prometa, rad objašnjava najčešće korištene komponente u arhitekturi kibernetičkih poligona i na koji način se odvijaju radi boljeg upoznavanja problematike. Kako bi generator prometa bio učinkovit potrebno je da autonomno generira podatke uz korištenje različitih protokola te da su generirani podaci naizgled nasumični i da ih je teško profiltrirati.

Cilj je proučiti razne pristupe rješavanju tog problema i u konačnici ponuditi optimalno rješenje, kako bi se to rješenje moglo iskoristiti u nekom od kibernetičkih poligona. Rješenje koje je pronađeno kao najkorisnije, ponaša se kao web pauk (engl. web spider) za pretraživanje po internetu. Nadalje, potrebno je napraviti simulaciju i vidjeti nalikuju li generirani podaci onome što možemo očekivati na nekoj lokalnoj mreži koja ima korisnike.

Rad je strukturiran tako da se u drugom poglavlju opisuju kibernetički poligoni kako bi se razradila problematika. Opisuju se razni korisnici prilikom odvijanja vježbe, te zatim infrastruktura poligona. Drugo poglavlje također razrađuje razne implementacije kibernetičkih poligona, od rješenja otvorenog koda do komercijalnih. Glavna tematika rada je pronalaženje adekvatnog generatora prometa za kibernetički poligon, pa se u trećem poglavlju razmatraju različiti pristupi. Svaki od pristupa ima svoje prednosti i mane, no samo neki zadovoljavaju karakteristike koje mora imati generator prometa. Nakon što je pronađen adekvatan alat za generiranje prometa, sljedeće potpoglavlje opisuje postupak simuliranja infrastrukture i opisuje dobivene rezultate. Zatim nad dobivenim podacima se dokazuje da su generirani podaci slični stvarnim podacima koje generira živa mreža. To je dokazano preko fraktalnih svojstava i samo-sličnosti. Rad završava zaključkom i popisom korištene literature.

2. Kibernetički poligoni

Povećavanjem kompleksnosti informacijskih sustava, povećava se i mogućnost izvršavanja napada na sustav i broj ranjivosti sustava. Iako su napori za osiguranjem od napada veliki, nikada se neće moći ukloniti sve ranjivosti sustava.

Prema tvrtki Herjavec Group, koja se bavi kibernetičkom sigurnošću, kibernetički napadi su najveća prijetnja svakoj tvrtki na svijetu [4]. Procjenjuju da će kibernetički napadi koštati svijet oko 6 trilijuna dolara do 2021. godine što je duplo više od štete koju su kibernetički napadi ostvarili do 2015. godine kada je procijenjena šteta na 3 trilijuna dolara. Također se prema njima većina učinjene štete odnosi na uništavanje bitnih podataka, krađa intelektualnog vlasništva, krađa novca i gubitak produktivnosti tvrtke.

Tvrtke zato ulažu u edukaciju osoblja na kibernetičkim poligonima. Kibernetički poligon je virtualno okruženje koje je sigurno za uvježbavanje osoblja. Poligoni sadrže primjere napada iz stvarnog svijeta iako se radi samo o simulaciji, cilj je da zaposlenici dobiju vještine koje će im možda trebati kako bi mogli obraniti informacijski sustav tvrtke.

Kibernetički poligoni nude brojne prednosti tvrtkama koje ulažu u treniranje osoblja. Prikazuju prijetnje koje su stvarne, ali su prijetnje unutar virtualnog okruženja pa se mogu više puta pokrenuti. Okruženje u kojem se izvodi poligon je izolirano i kontrolirano te ne predstavlja prijetnju postojećoj infrastrukturi. Polaznici poligona lakše prepoznaju prijetnje i prilikom prepoznavanja prijetnje brže i efikasnije reagiraju na njih. Prednost poligona je što na praktičan način obučavaju polaznike za razliku od samog podučavanja teorije.

2.1. Korisnici kibernetičkog poligona

Korisnici kibernetičkog poligona se dijele u razne vrste timova kako bi se što vjernije simulirao scenarij napada kakav se može očekivati i u stvarnosti. Iako postoje brojni timovi u kibernetičkim poligonima, dva tima su uvijek nužna, crveni i plavi tim. Ostali timovi nisu nužni ili mogu biti automatizirani. Služe kako bi pomogli timovima ili omogućili izvršavanje vježbe. Slijedi popis i objašnjenje najbitnijih timova prilikom izvođenja vježbe na kibernetičkim poligonima [5].

- **Crveni tim**

  Crveni tim predstavlja napadača. Njihova glavna zadaća je da pronađu sigurnosne nedostatke u infrastrukturi i da ih iskoriste kako bi izvršili napad. Svrha je da ljudi koji su u tom timu kasnije imaju sposobnost razmišljanja kao napadač i da to znanje mogu primijeniti prilikom penetracijskog testiranja vlastitih sustava.

- **Plavi tim**

  Skupina ljudi koja se brani od napada kojeg vrši crveni tim. Plavi tim treba identificirati područje napada koje bi crveni tim mogao napasti i spriječiti ih u tom naumu. Ako je

crveni tim već iskoristio sigurnosnu manu tada ih aktivno pokušavaju spriječiti kako ne bi napravili daljnju štetu. Korisno je za ljude koji održavaju infrastrukturu i trebaju imati vještine s kojima bi se branili od napadača.

- **Bijeli tim**

  Bijeli tim oblikuje vježbu, scenarij kojim će se odvijati napad, ciljeve i pravila. Postavljaju pravila crvenom i plavom timu, postavljaju sigurnosne prijetnje u virtualno okruženje. Tijekom cijele vježbe prate što se događa unutar okruženja i što rade obje strane. Mogu pružiti pomoć pojedinom timu ako je to nužno.

- **Zeleni tim**

  Zeleni tim je zadužen za razvoj okoline u kojoj se odvija vježba. Također održavaju infrastrukturu funkcionalnom i prate njeno stanje tijekom vježbe. Ako se pojavi neka greška unutar okoline ili neki tim ne namjerno onesposobi jednu od komponenti, zeleni tim uklanja štetu.

- **Narančasti tim**

  Narančasti tim kao ulogu ima zadavanje zadataka plavom timu tijekom vježbe. Ako plavi tim riješi zadatke koje im je zadao narančasti tim, dobivaju bodove za zadatak.

- **Ljubičasti tim**

  Ljubičasti tim je zadužen za prenošenje informacija između crvenog i plavog tima koji se natječu. Pomoću dobivenih informacija se povećava efektivnost oba tima. Crveni tim dobiva bitne informacije gdje mogu izvršiti napad, dok plavi tim ima dobiva informacije o tome gdje crveni tim vrši napad.

- **Žuti tim**

  Žuti tim simulira korisnike infrastrukture. Mogu koristiti dijelove infrastrukture kao što je generator prometa ili neke druge alate za koje generator prometa nije primjenjiv. Mrežni promet koji generiraju može služiti kao naputak crvenom ili plavom timu, gdje informacija ukazuje što se može iskoristiti za napad ili obranu.

2.2. Arhitektura kibernetičkog poligona

Virtualno okruženje je najbitnije prilikom održavanja vježbe. Razlog zašto se vježbe ne održavaju na već postojećoj infrastrukturi je to što nije poželjno da se u stvarnu mrežu ugrade zlođudni programi, onesposobi rad usluga ili da se ukradu stvari i potencijalno povjerljivi podaci. Kibernetički poligon mora imati različite komponente kako bi vježba izgledala što realnije. Sljedeće točke opisuju nužne komponente kibernetičkih poligona [6].

- **Virtualna mreža**

  Virtualna mreža treba oponašati stvarnu mrežu. Cilj virtualne mreže je da bude sigurno okruženje za uvježbavanje, što lakša za održavanje, a uz to da što zornije imitira komponente i događaje na stvarnoj mreži. Potrebno je da simulirana čvorove, preklopnike i premosnike.

- **Napadački mehanizam**

  Napadački mehanizam je nužan kako bi se u virtualnoj mreži pokrenuli razni kibernetički napadi. Potrebno je da podržava razne vrste napada kao što su DDoS napad, napad pomoću ucjenjivačkog koda, krađu podataka. Neki poligoni automatiziraju rad napadačkog mehanizma, dok drugi samo osiguravaju pristup potrebnim alatima crvenom timu.

- **Generator prometa**

  Generator prometa treba oponašati benigni promet i proizvoditi šum unutar virtualne mreže kako bi se otežalo otkivanje napadačkih paketa.

- **Virtualni SOC**

  Virtualni SOC (engl. Security Operations Center) je virtualni sigurnosni operacijski centar. Virtualni SOC je alat kojem se može pristupiti pomoću internetskog preglednika i služi kako bi se lakše mogli pratiti sigurnosni događaji unutar sustava u stvarnom vremenu. Pruža potrebne informacije plavom timu kada se dogodio napad i na kojoj komponenti. Cilj komponente je da plavi tim što brže reagira na napad.

2.3. Kibernetički poligoni otvorenog koda

Kibernetički poligoni su kompleksne infrastrukture i zahtijevaju redovito održavanje kako bi prijetnje bile u skladu s vremenom. Zbog toga se većina kibernetičkih poligona plaća i u vlasništvu su velikih tvrtki. Postoji par projekata otvorenog koda koji svima omogućuju uvježbavanje na kibernetičkim poligonima.

2.3.1. OCCP

OCCP (engl. Open Cyber Challenge Platform) [7] je kibernetički poligon otvorenog koda. Sastoji se od VSN (engl. Virtual Scenario Network) komponente koja predstavlja mrežu virtualnog scenarija. Podržava crveni i plavi tim koji sudjeluju u vježbi. Sivi tim predstavljaju skripte koje generiraju korisnički promet i bijeli tim su ili ljudi ili skripte koji prate sustav. OCCP podržava različite vrste vježbi, neke od njih su opisane u nastavku.

- Mrežna obrana gdje su korisnici poligona u plavom timu, a crveni tim imitiraju skripte. Plavi tim dobiva bodove za sve usluge koje nastave neometano raditi, a gube bodove za podatke koje crveni tim ukrade ili ako usluge u virtualnom okruženju budu onemogućene.
- Penetracijsko testiranje (engl. penetration testing) kada su korisnici u crvenom timu, a plavi tim imitiraju skripte. Korisnici nastoje naći mane u sustavu i ukrasti podatke i dobivaju bodove za ukradene podatke i sve usluge kojima onemoguće rad.
- Digitalna forenzika kada korisnici traže tragove napada unutar virtualnog okruženja. Crveni tim može i ne mora nužno sudjelovati u ovoj vježbi.
- Sigurno programiranje gdje plavi tim razvija programe koje kasnije crveni tim koji glume skripte, pokušava iskoristiti različitim napadima kao npr. napad ubacivanjem SQL izraza.

- Reagiranje na incident gdje skripte koje predstavljaju crveni tim glume napadača, a korisnici poligona nastoje istražiti koji podaci su ukradeni i nastoje saznati identitet napadača.
- Analiza zloćudnog koda gdje korisnici pripadaju plavom timu i nastoje locirati zloćudan kod i napraviti analizu njegovog utjecaja na sustav.

Svaka vrsta vježbe ima različite podvrste koji se nazivaju scenarijima. Svaki scenarij je potrebno postaviti u virtualno okruženje prije početka vježbe. Poligon se sastoji od dvije bitne komponente, igraćeg poslužitelja (*engl.* Game server) i nadzornog virtualnog okruženja (engl. Administrative VM).

Igraći poslužitelj je virtualni stroj koji pokreće scenarij. Prilikom pokretanja igraći poslužitelj pokreće generator prometa i skripte koje glume crveni ili plavi tim. Prati bodove koje su polaznici poligona ostvarili i vrijeme trajanja. Bijeli tim pomoću igraćeg poslužitelja komunicira s igračima. U internetskom pregledniku omogućuje praćenje trenutačnog stanja scenarija.

Administrativno virtualno okruženje pokreće igraći server i virtualnu mrežu koja simulira infrastrukturu. Pogodno je što OCCP nudi pakete scenarija koji sadrže dokumentaciju vježbe, informaciju o potrebnim tehničkim vještinama polaznika i opise svih komponenti i skripti te kako rukovati njima. Daljnji razvoj OCCP alata je obustavljen te korisnička podrška više nije aktivna, ali korisnici i dalje mogu preuzeti izvorni kod i nastaviti daljnji razvoj alata.

2.3.2. AWS CyberRange

CyberRange [8] je projekt koji predstavlja nacrt (engl. blueprint) kibernetičkog poligona otvorenog koda. Projekt sadrži podršku za obrambene mehanizme, napadačke alate, alate za reverzno inženjerstvo i razne druge. CyberRange nudi brojne sustave sa sigurnosnim manama i alate koji su potrebni za penetracijsko testiranje tih sustava.

CyberRange koristi brojne moderne tehnologije kao što je opisano u tablici 2.1.

Tehnologija	Objašnjenje tehnologije
Amazon Web Services [24]	sustav koji omogućava upravljanje sustavima na oblaku
Kali Linux [25]	distribucija operacijskog sustava Linux za penetracijsko testiranje
Nessus [26]	alat koji traži sigurnosne mane
Commando-VM [27]	virtualno okruženje za penetracijsko testiranje na Windows operacijskom sustavu
Terraform [28]	alat otvorenog koda koji spada u kategoriju 'infrastruktura kao kod' gdje se infrastruktura prikazuje pomoću konfiguracijskih datoteka
Docker [29]	alat koji koristi virtualizaciju na razini operacijskog sustava i pruža uslugu unutar okruženja zvanog kontejner
DetectionLab [30]	virtualno okruženje dizajnirano za obučavanje ljudi kako bi se znali obraniti od napada.
Inspec [31]	alat otvorenog koda za testiranje infrastrukture i uočavanje mogućih propusta.

Tablica 2.1. Korištene tehnologije CyberRange alata

Koriste se još mnoge druge tehnologije. Cilj projekta je da se korisnicima jednostavno omogući okruženje za napad ili obranu.

2.4. Komercijalni kibernetički poligoni

Kibernetičke poligone je zbog njihove kompleksnosti teško za održavati. Potrebno je konstantno voditi računa da poligoni sadrže najnovije vrste napada. Radi kompleksnosti poligona, potrebna je podrška tvoraca poligona kako bi pomogle tvrtkama organizirati vježbu i otkloniti moguće tehničke poteškoće prilikom postavljanja okruženja kibernetičkog poligona. Nažalost, radi karaktera otvorenog koda, teško je očekivati takvu vrstu potpore i redovite nadogradnje te je zato većina tvrtki je primorana koristiti komercijalna rješenja. Ostatak poglavlja ukratko objašnjava najpoznatije komercijalne kibernetičke poligone.

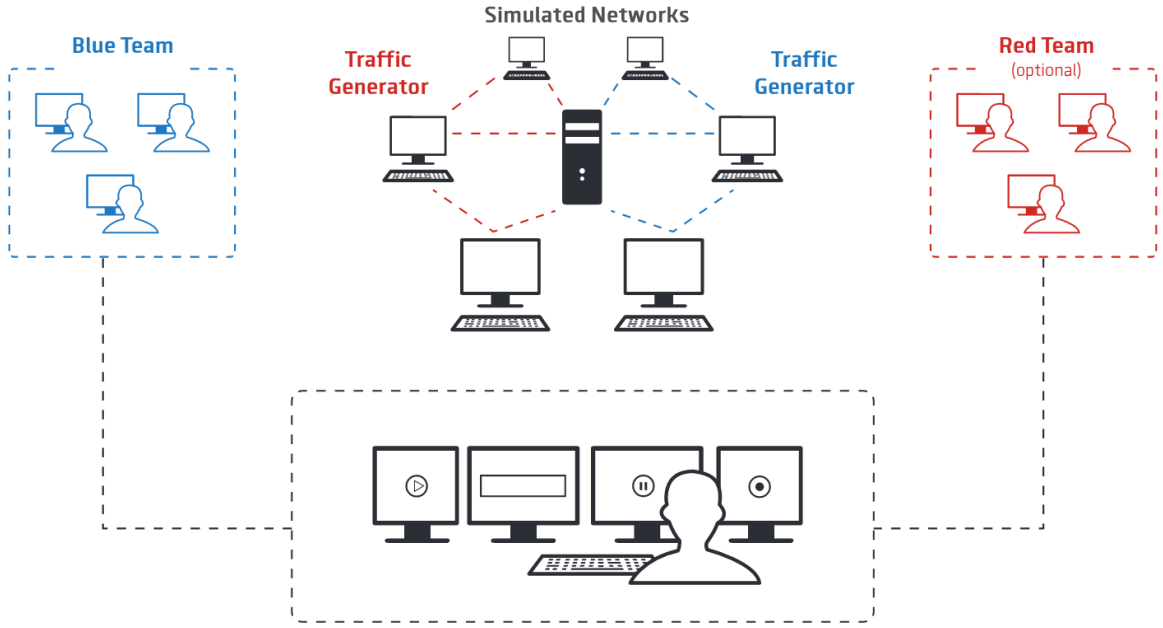
2.4.1. Cyberbit

Cyberbit [6][9] nudi rješenje kibernetičkog poligona. Infrastruktura je izdrživa i fleksibilna. Kibernetički poligon omogućuje promjene na scenarijima prema potrebama. Okruženje unutar scenarija automatski kreira benigni korisnički promet, napadačke radnje na mreži čime se smanjuje potreba za crvenim timom. Ako postoji potreba za izradom vlastite inačice scenarija, organizatori vježbe ga mogu kreirati samostalno. Poligon se može izvršavati na oblaku ili u drugim okruženjima. Slika 2.2 prikazuje arhitekturu spomenutog poligona.

Poligoni podržavaju treniranje korisnika podijeljenih u crveni i plavi tim, ali je moguće i individualno uvježbavanje. Individualni trening omogućava prilagođavanje scenarija specifičnim potrebama korisnika te usavršavanje u posebnim vještinama. Podržan je i CTF (*engl.* Capture The Flag) način rada gdje korisnici dobiju zadatak za riješiti i cilj je dobiti rješenje koje se naziva zastavicom (engl. flag) [32]

Virtualno okruženje ovog kibernetičkog poligona je pogodno za razne vrste penetracijskog testiranja, a i za istraživanje zloćudnog koda i raznih vrsta napada. Korisnici mogu dobiti pregled u način rada različitih vrsta napada.

Platforma je pogodna za podučavanje studenata gdje profesori mogu zadavati studentima kolegija zadatke preko platforme i pratiti njihov napredak tijekom kolegija.



Slika 2.2 Arhitektura Cyberbit kibernetičkog poligona [9]

2.4.2 Cisco Cyber Range

Cisco je tvrtka poznata u informacijskoj tehnologiji, mrežnim i kibernetičkim rješenjima [10]. Cisco Cyber Range je kibernetički poligon koji je razvila tvrtka Cisco. Poligon podržava preko 50 različitih scenarija napada na različitim tehnologijama. Unutar virtualnog okruženja su pokrenute stotine aplikacija s 200-500 zloćudnih programa koji predstavljaju sigurnosnu prijetnju koju je potrebno otkloniti. Virtualno okruženje simulira žičani i bežični pristup. Kibernetički poligon sadržava detaljno razrađenu simulaciju mreže. Također sadržava simulator klijenata, poslužitelja i aplikacija. Cisco Cyber Range koristi generator prometa kao i svi drugi poligoni. Scenariji podržavaju napade najnovijih prijetnji, DDoS napade, napade na aplikacije, krađa podataka, zloćudni kodovi na mobilnim uređajima i računalima i brojne druge. Kibernetički poligon se odvija kao radionica koju se pohađa 3 – 5 dana što čini ovaj kibernetički poligon jedinstvenim.

3. Generiranje benignog korisničkog prometa

Kibernetički poligon je virtualno okruženje, kao što je to ranije opisano. Sav promet unutar virtualnog okruženja će biti promet kojeg razni alati autonomno generiraju, no nedostaje šum kojeg stvaraju korisnici. Kako bi poligoni što više nalikovali stvarnim okruženjima, potrebno je u kibernetički poligon integrirati generator prometa. Generatora prometa bi unutar poligona trebao proizvoditi šum koji bi bio nalik onomu kojeg proizvode korisnici unutar mreže. Isto tako je vrlo bitno da bez generatora prometa, napadačka ili obrambena strana može lako uočiti sav promet koji generira druga strana. Ideja generatora prometa je da stvara promet unutar mreže kibernetičkog poligona koji nalikuje prometu na stvarnoj mreži. Bitna je značajka generatora prometa da generira podatke koji nalikuju korisničkim tj. teško je predvidjeti takav promet. Treba koristiti široki spektar protokola i u različitim frekvencijama generirati promet. Tim uvjetom osiguravamo da se uzorak što teže pronađe i što teže razazna od stvarnih podataka.

Postoje brojna komercijalna rješenja ovog problema. Jedno od rješenja je ponudbeno od tvrtke Ixia koja je napravila proizvod zvan IxNetwork [33]. IxNetwork generira promet raznih vrsta protokola te može oponašati korisnike aplikacija. Cilj poglavlja je proučiti razne alate otvorenog koda koji bi se mogli koristiti kao generator prometa, identificirati njihove prednosti i mane te u konačnici odabrati jedan alat koji ima potrebna svojstva. Potrebno je na primjeru utvrditi da alat uistinu ima potrebna svojstva i zato je dio poglavlja posvećen testiranju alata. Nakon što se alat testira i zabilježi generirani promet, provodi se analiza nad tim podacima i uspoređuju se svojstva sa stvarnim mrežama. Jedno od svojstava je fraktalnost mreža te je kraj poglavlja posvećen proučavanju svojstva fraktalnosti generiranog prometa.

3.1. Generiranje prometa

Puno alata se može koristiti kao generator prometa. Kako bi se pronašao adekvatan alat za generiranje prometa potrebno je proučiti razne alate i vidjeti koje su njihove prednosti i mane. Velik broj tih alata služi za testiranje stabilnosti i kapaciteta mreže, jer mogu odaslati puno paketa u kratkom vremenu. Problem kod većine tih alata je da, iako generiraju podatke, podaci su jako slični i filtriranje tih podataka je izrazito jednostavno. Bitno je također da alat ne zahtijeva puno resursa za svoj rad, jer inače smanjujemo kvalitetu i kapacitete virtualnog okruženja. Ostatak potpoglavlja opisuje alate koji generiraju promet i moguće ih je djelomično primijeniti unutar kibernetičkog poligona kao generatore prometa.

Ostinato

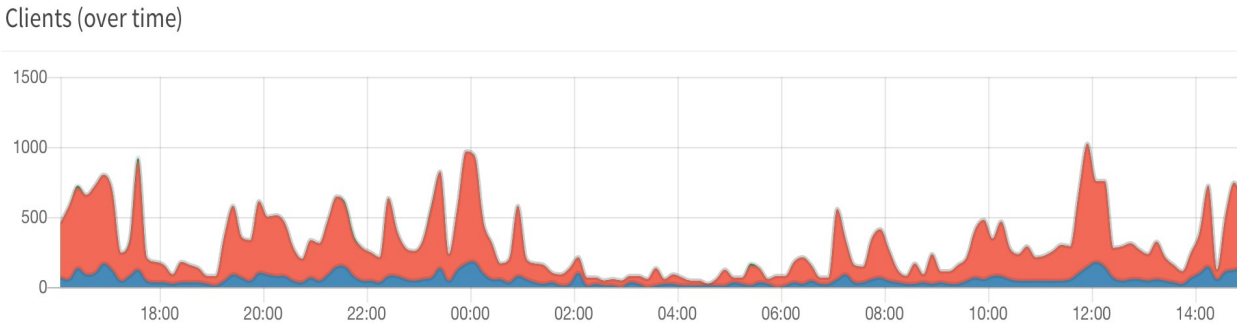
Alat Ostinato [11] je alat otvorenog koda iako ima naprednija komercijalna verzija. Ostinato se koristi kao sastavljač paketa i generator prometa. Alat se pretežno koristi za testiranje mrežnih sustava. Ostinato je fleksibilan alat te može generirati razne vrste prometa, koliko često se taj promet generira, koliko paketa će se generirati i koji protokoli će se koristiti. Podržava brojne

protokole kao npr. VLAN, ARP, IPv4 i IPv6, IP, TCP i UDP, ICMPv4 i ICMPv6, IGMP, MLD, IEEE 802.3 LLC, SNAP i protokole tekstualnog tipa. Alat pruža brojne mogućnosti kojima bi se mogla testirati mreža ili programske funkcionalnosti. Iako podržava široki spektar protokola, ne zadovoljava bitno svojstvo generatora prometa, a to je nepredvidivost. Ukoliko bi se koristio Ostinato kao generator prometa, moguće je definirati puno različitih paketa koje će slati, ali što bi se duže koristio to bi se sve više i više vidjeli uzorci u ponašanju paketa i u konačnici bi se moglo predvidjeti ponašanje.

DNoiSe

DNoiSe [12] je jednostavan alat otvorenog koda koji generira nasumičan DNS promet. Alat promatra aktivnost mreže te generira DNS promet tako da sintetički generirani promet ne odstupa previše od uobičajenih aktivnosti mreže.

Na slici 3.1 plavom bojom je označen DNS promet koji je generiran pomoću DNoiSe alata, a crvenom bojom stvarni promet kojeg generira korisnik. Na grafu je vidljivo da sintetički promet ne odstupa previše od prometa kojeg je generirao korisnik, te smanjuje šansu da se otkrije što je sintetički promet. Alat je napravljen s ciljem generiranja šuma kako bi DNS poslužitelji teže profilirali korisnika alata. Alat prati aktivnost mreže te generira dodatnih 10% DNS paketa. Odabir koji DNS paket će biti sljedeći generiran je nasumičan i bira se iz Cisco liste koja sadrži milijun najpopularnijih domena [13]. DNoiSe generira nepredvidive podatke, ali ne koristi puno protokola i problem je što zahtjeva od korisnika da generira neki promet. DNoiSe alat radi toga nije pogodan kao generator prometa.



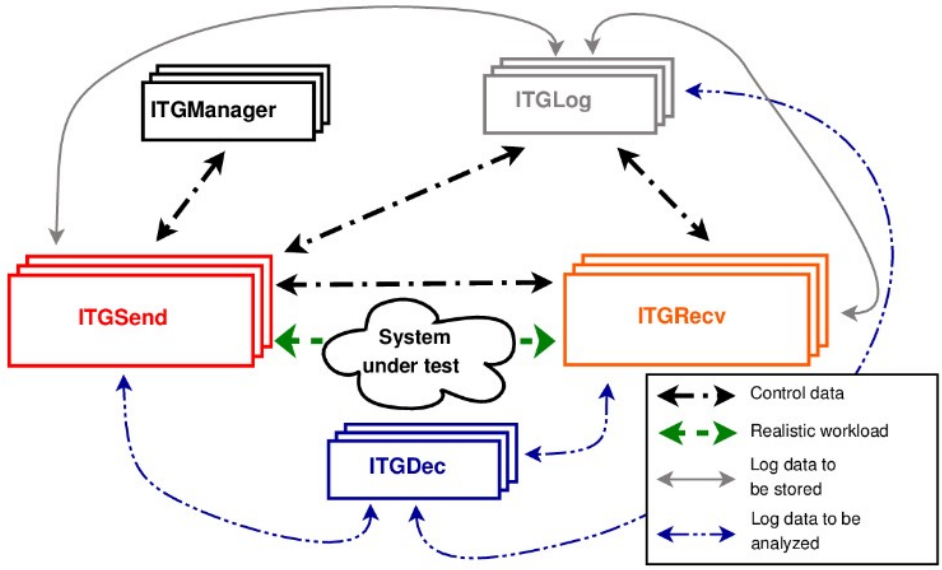
Slika 3.1 Generirani DNS promet [13]

D-ITG

D-ITG (engl. Distributed Internet Traffic Generator) [14] je alat koji može generirati IPv4 i IPv6 promet. Također, sadrži alate kojima korisnik može mjeriti performanse mreže kao na primjer propusnost, kašnjenje i gubitak paketa. Alat može generirati promet koji prati stohastički model za veličine paketa i razmak između slanja paketa (engl. inter departure time) paketa kako bi oponašao protokole aplikacijskog sloja. Od protokola transportnog sloja podržava TCP, UDP, SCTP i DCCP protokole. Ima kompleksniju arhitekturu od prethodno navedenih alata.

Arhitektura D-ITG alata je sačinjena od 4 bitne komponente koje su prikazane na slici 3.2. ITGSend komponenta koja je zadužena da generira promet i prosljeđuje ga ITGRecv

komponenti. ITGRecv komponenta služi kako bi primala tokove paketa od strane ITGSend komponente. Komponenta ITGSend je višedretvena, može slati više paketa prema jednoj ili više instanci ITGRecv komponente. Isto tako ITGRecv može primati pakete poslane s više ITGSend komponenti. ITGLog komponenta prima podatke i zapisuje sve aktivnosti ITGSend i ITGRecv komponenta u log datoteku. ITGDec dekodira i radi detaljnu analizu log datoteka koje su generirane tijekom eksperimenta. Podaci koji se generiraju mogu se uvelike razlikovati jer podržava generiranje paketa koje imitira podatke koje generiraju razne aplikacije npr. Quake3 i CounterStrike igre, Telnet, DNS i VoIP. Problem ovog alata je što je kompleksan i teško je napraviti ispravnu konfiguraciju mreže.



Slika 3.2 Arhitektura D-ITG alata

Postoje razni programi koji mogu generirati razne vrste prometa. Problem je zadovoljavanje dvaju pravila. Brojni alati poput Ostinato alata, zahtijevaju prethodno definiranje paketa što znači da će se u konačnici moći naći uzorak kojim se generiraju podaci. Alati poput DNoIse alata su napravljeni kako bi zaštitili korisnika računala od pružatelja internet usluga, tj. kako bi im otežali prikupljanje podataka o korisniku. Problem kod takvih alata je da je potrebno da korisnik računala generira podatke što je beskorisno za slučaj kibernetičkog poligona, jer želimo da generator prometa autonomno generira podatke.

3.2. Alat Noisy

Noisy [15] je jednostavan alat otvorenog koda koji pruža mogućnost generiranja nasumičnog prometa. Inspiriran je projektima poput web-traffic-generator [16] koji zapravo spadaju u kategoriju "pauka" (engl. crawler) [17].

Pauci su programi koji automatski pretražuju brojne dokumente ili stranice na internetu u cilju indeksiranja sadržaja. Web tražilice koriste alat pauk kako bi indeksirale stranice na internetu. Indeksiranje stranica je nužan dio za rad tražilica kako bi mogli stranici dodijeliti npr. kategoriju kojoj pripada ili termine koji se najčešće pojavljuju na toj stranici. Pauk sprema stranice u repozitorije te na temelju sadržaja stranice stvara indekse koje kasnije tražilice koriste kako bi korisnicima pružile kvalitetniju uslugu pretraživanja. Svaki pauk kada posjeti stranicu treba dohvatiti poveznice na druge stranice iz HTML datoteke koju je primio. To je nužno kako bi mogao znati koju će iduću stranicu posjetiti, jer bi u suprotnom samo prestao s pretraživanjem. Svakom pauku je potrebno dati početan skup stranica koji se naziva *sjeme* (engl. seed). Kada pauk započinje s radom tada prvo posjeti jednu od stranica iz sjemena. Iz posjećene stranice izlučuje sve poveznice prema drugim stranicama i sprema ih u skup sjemena i zatim rekurzivno ponavlja pretraživanje sve dok ima neposjećenih stranica u skupu sjemena. Tako pauk može u nedogled pretraživati internetom i raditi neki predodređeni zadatak kao što je prikupljanje podataka ili indeksiranje stranica. Pauci se također mogu iskoristiti kako bi se generirao šum.

Noisy je jednostavna implementacija pauka. Alat je implementiran u programskom jeziku Python te generira nasumičan HTTP, HTTPS i DNS promet. Cilj alata je kada pružatelji internetske usluge prikupljaju podatke o korisniku, učiniti prikupljene podatke irelevantnima, jer će biti pomiješani s velikim brojem podataka koje korisnik nije generirao i zapravo predstavljaju šum.

Alatu je potrebno u naredbenom retku predati putanju do konfiguracijske datoteke koja je prikazana u ispisu 3.1. U konfiguracijskoj datoteci se mogu definirati brojni parametri koji utječu na rad alata Noisy te tako omogućavaju veliku fleksibilnost. Kao i kod svakog pauka potrebno je definirati početan skup stranica od kojih će on početi pretraživati, tako zvano sjeme koje se postavlja unutar "root_url" liste u konfiguracijskoj datoteci. Prilikom pokretanja, alat će odabrati nasumice jednu od tih stranica i posjetiti je.

Svakoj poveznici se pridjeljuje dubina na kojoj se nalazi. Ako želimo ograničiti dubinu do koje će pauk pretraživati tada je potrebno postaviti "max_depth" vrijednost u konfiguracijskoj datoteci. Kada se dođe do dubine koja je definirana u konfiguracijskoj datoteci, program ne proširuje skup poveznica s poveznicama koje se nalaze na stranici s maksimalnom dubinom.

Kako bi podaci izgledali što više nasumični, ne pretražuje se uvijek u istom vremenskom odmaku. Naime postavljaju se vrijednosti dvije varijable "min_sleep" i "max_sleep" koje određuju vrijeme koje će generator prometa čekati i ne raditi ništa. Vrijeme čekanja također je nasumično, jer se uzima jedan nasumičan broj u intervalu između te dvije granice koji se zatim koristi kao vrijeme čekanja.

Konfiguracijska datoteka omogućava definiranje skupa stranica ili datoteka koje ne želimo nikada posjećivati, jer nam ne pridonose u generiranju prometa ili potencijalno mogu biti opasne i ugroziti računalo koje pokreće alat.

Skup podataka koji se nalazi u datoteci i unosi dodatnu nepredvidivost generiranim podacima je polje podataka "user_agent". Svi podaci unutar tog skupa podataka se mogu koristiti kako bi se postavilo polje unutar zaglavlja HTTP paketa. Podaci unutar „user_agent” polja predstavljaju identitet aplikacije, operacijskog sustava i podatke o procesoru. Ti podaci se postavljaju u zaglavlje HTTP paketa kojim se šalje zahtjev za paketima od poslužitelja. Ova funkcionalnost, kombinirana s različitim intervalima kada se pretražuje garantira veliki skup različitih podataka. Ovakav tip generatora prometa je potrebno samo pokrenuti i ostaviti da pretražuje mrežom i alat će se lažno predstavljati kao razni korisnici. Velika prednost je u tome što se ponaša jako

nepredvidivo, na početku se može predvidjeti na kojim će stranicama pretraživati, no nakon što prestane pretraživati po stranicama početnog skupa postaje nepredvidiv. Alat je jednostavno postaviti zato što se radi samo o jednoj komponenti i potreban je samo Python za izvršavanje. Ne zahtijeva puno resursa za rad te time ne ometa rad ostalih bitnih komponenti unutar kibernetičkog poligona. Iz konfiguracijske datoteke je izostavljen skup podataka "user_agents" jer je podataka izuzetno puno.


```
{
  "max_depth": 25,
  "min_sleep": 0,
  "max_sleep": 1,
  "timeout": false,
  "root_urls": [
    "http://4chan.org",
    "https://www.reddit.com",
    "https://www.yahoo.com",
    "http://www.cnn.com",
    "http://www.ebay.com",
    "https://www.facebook.com",
    "https://wikipedia.org",
    "https://youtube.com",
    "https://github.com",
    "https://www.ebay.com",
    "https://www.twitch.tv",
    "https://www.aliexpress.com",
    "https://www.linkedin.com",
    "https://twitter.com",
    "https://medium.com",
    "https://thepiratebay.org"
  ],
  "blacklisted_urls": [
    "https://t.co",
    "t.umblr.com",
    "messenger.com",
    "itunes.apple.com",
    "l.facebook.com",
    "bit.ly",
    "mediawiki",
    ".css",
    ".ico",
    ".xml",
    "intent/tweet",
    "twitter.com/share",
    "dialog/feed?",
    ".json",
    "zendesk",
    "clickserve",
    ".png",
    ".iso"
  ]
```


Ispis 3.1 Konfiguracijska datoteka alata

3.3. Simulacija mreže kibernetičkog poligona

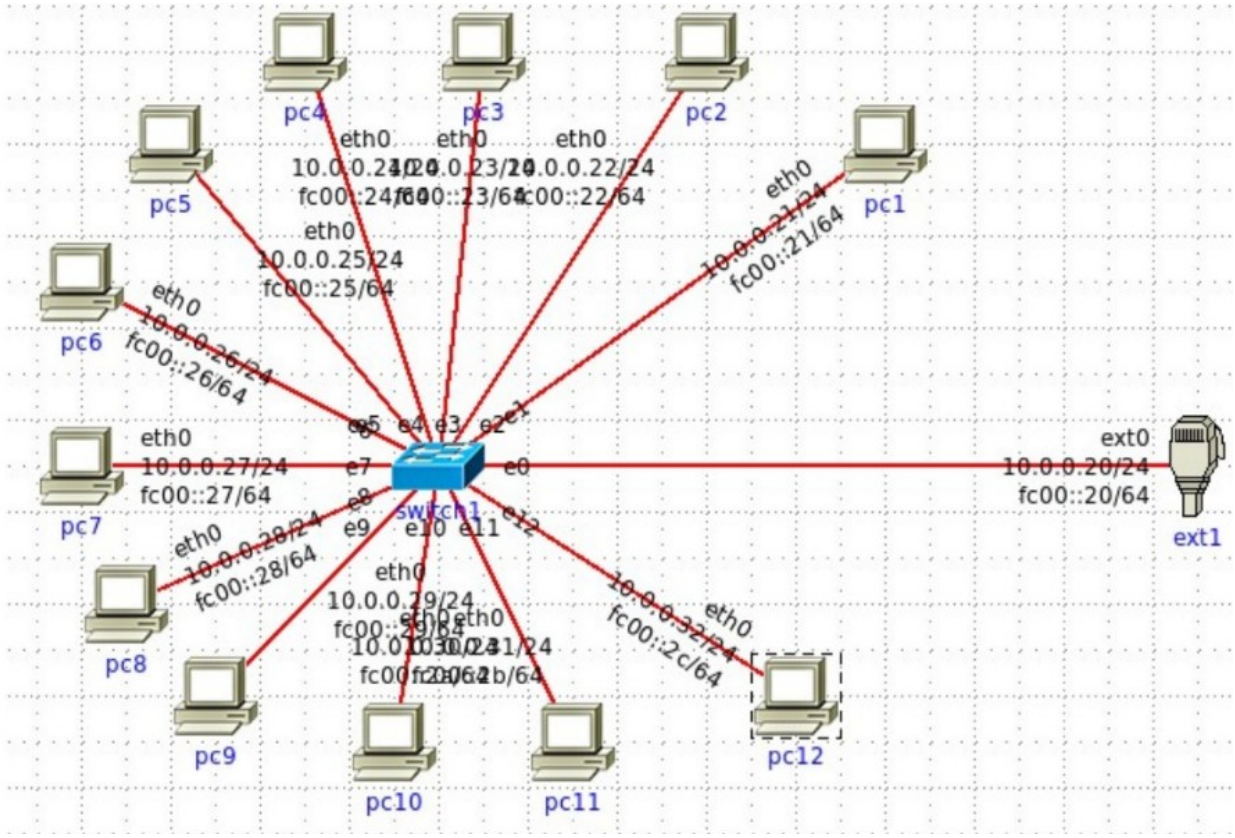
Kako bi se testirao alat i dokazala njegova valjanost potrebno je napraviti simulaciju mreže i podesiti okruženje kako bi alat mogao generirati promet. U vremenu kada alat generira promet potrebno je i snimiti sav promet kako bi se mogla provjeriti valjanost generiranih podataka. Virtualno okruženje kibernetičkog poligona će biti izolirano i eventualno imati pristup vanjskom internetu. Postoje razni alati koji bi se mogli koristiti za simulaciju mreže kao na primjer Docker, no alat Imunes nudi bolju simulaciju stvarnih računala zato što nudi razne vrste komponenti, jednostavnije je za postavljanje i ujedno fleksibilnije od Docker alata.

Imunes (engl. Integrated Multiprotocol Network Emulator/Simulator) [18] je alat otvorenog koda razvijen na Fakultetu elektrotehnike i računarstva u Zagrebu. Alat služi sa simuliranje mrežnih topologija i pokreće se na operacijskom sustavu Linux. Imunes alat može u stvarnom vremenu simulirati IP mreže. Simulira veliki broj virtualnih čvorova na jednom fizičkom stroju gdje svaki čvor može pokretati različite Unix programe. Jedna od bitnih prednosti alata je da je izrazito brz i mrežne topologije su prenosive. Imunes alat je korišten kako bi se simulirala mreža kojom na početku ne teku paketi, ali integriranjem alata Noisy se može simulirati mreža koja je nalik stvarnoj mreži s korisnicima.

Korištena je inačica Imunes alata koja je postavljena u virtualnoj slici baziranoj na operacijskom sustavu FreeBSD. Unutar grafičkog sučelja potrebno je postaviti različite komponente kako bi kreirali željenu topologiju. Potrebna je komponenta „PC” koja predstavlja računalo kojim korisnik pretražuje po internetu. Komponenta PC je element mrežnog sloja koji ne prosljeđuje pakete i ima statičke staze. PC za razliku od drugih komponenti mrežnog sloja ne pokreće mrežne usluge. Komponenta PC u simulaciji treba biti spojena s vanjskim internetom i služi kako bi u toj komponenti pokrenuli alat Noisy. Prije pokretanja alata potrebno je postaviti sve potrebne pomoćne alate kako bi se Noisy mogao biti pokrenut. Tada svako od računala predstavlja jednog od korisnika mreže i svaki korisnik generira promet, neovisno o drugim računalima.

Sva računala unutar topologije moramo prije svega povezati preko LAN preklopnika kako bi dobili lokalnu mrežu s više računala. LAN preklopnik je ostvaren preko komponente „LAN switch” koja je element fizičkog sloja koji prosljeđuje dolazeće pakete na povezane čvorove.

Mreža unutar alata Imunes i dalje nije povezana na vanjski internet. Kako bi lokalna mreža imala pristup internetu prvo je potreban internetski pristup na računalu u kojem se pokreće alat Imunes. Potrebno je dodati komponentu „External connection“ koja omogućuje spajanje virtualnog čvora sa sučeljem na stvarnom računalu i zatim povezati je s LAN preklopnikom čime se omogućuje pristup internetu unutar Imunes okoline. Na topologiji je kreirano 12 računala gdje svako generira promet i komunikacija se ostvaruje preko sučelja vanjske poveznice kao što je prikazano na slici 3.3.



Slika 3.3 Topologija korištene mreže u alatu Imunes

Pokretanjem topologije, komponenta „External connection“ stvara novo sučelje na računalu koje pokreće Imunes. Kako bi novo sučelje imalo pristup internetu potrebno je napraviti mrežni premosnik. Premosnik je virtualno sučelje koje spaja dvije odvojene mrežne komponente i omogućuje njihovu komunikaciju. Premosnik je nužan kako bi se povezalo sučelje koje je kreirao Imunes sa sučeljem na računalu koje pokreće alat Imunes. Operacijski sustav FreeBSD ima ugrađen skup alata koji omogućuju stvaranje i spajanje komponenti na premosnik.

Prvo je potrebno stvoriti novi premosnik naredbom:

ifconfig bridge create

Zatim je potrebno spojiti dva sučelja kojima želimo omogućiti komunikaciju preko premosnika:

ifconfig bridge0 addm em0 up
ifconfig bridge0 addm "ime Imunes sučelja" up

Pomoću navedene tri naredbe, novo kreirano sučelje ima pristup vanjskom internetu. Računala u lokalnoj mreži i dalje nemaju pristup internetu zato što im nije dodijeljena IP adresa. Kako bi komponenti PC unutar Imunes simulacije bila dodijeljena IP adresa potrebno je pokrenuti Bash ljusku i zatražiti IP adresu od DHCP poslužitelja. Zahtjev za IP adresom od DHCP poslužitelja se postiže naredbom:

dhclient eth0

Kako bi računala počela s generiranjem podataka potrebno je postaviti okruženje za rad alata i pokrenuti ga. Sljedeća skripta postavlja alate potrebne za Noisy i pokreće Noisy nakon što ga dobavi:

```
python3.6 -m ensurepip –default-pip
pip install requests
curl -LO https://github.com/1tayH/noisy/archive/master.zip
unzip master.zip
cd nosiy-master/
python3.6 noisy.py --config config.json
```

Sav promet koji teče vanjskom poveznicom se snima. Za snimanje i jednostavnu analizu podatka je korišten alat WireShark. Tablice 3.1 i 3.2 su dobivene pomoću alata WireShark.

Tablica 3.1 prikazuje da je 42,44% paketa veliko između 40-80 okteta. Velik broj paketa se odašilje prilikom sinkronizacije i drugih procesa koji ne zahtijevaju veliku količinu podataka. Isto tako oko 43% podataka su paketi koji su veličinom blizu vrijednosti MTU koja postavljena na 1514 na računalnim komponentama u Imunesu. Podaci su u skladu s očekivanim, slični su podacima koji se mogu očekivati kada analiziramo promet koji je generirao korisnik pretraživanjem na internetu.

Tema / Vrijednost	**Broj paketa**	**Prosjek**	**Minimalna vrijednost**	**Maksimalna vrijednost**	**Frekvencija (ms)**	**Postotak**
Veličina paketa	141825	715,29	42	1514	0,1064	100%
0-19	0	-	-	-	0,0000	0,00%
20-39	0	-	-	-	0,0000	0,00%
40-79	60187	57,60	42	79	0,0451	42,44%
80-159	5367	107,11	80	159	0,0040	3,78%
160-319	6101	245,24	160	319	0,0046	4,30%
320-639	4588	436,91	320	639	0,0034	3,23%
640-1279	3442	948,64	640	1279	0,0026	2,43%
1280-2559	62140	1458,61	1280	1514	0,0466	43,81%
2560-5119	0	-	-	-	0,0000	0,00%
5120 i više	0	-	-	-	0,0000	0,00%

Tablica 3.1 Veličine generiranih paketa

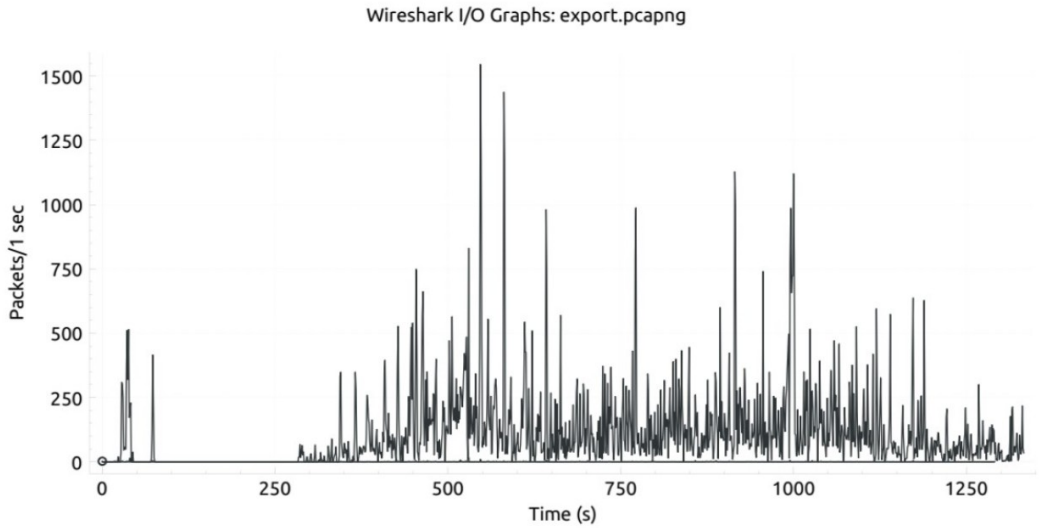
Tablica s podacima o veličini paketa pokazuje da se generiraju različite veličine paketa, te da se ne može naći uzorak kojim bi mogli profiltrirati stvarni od generiranog prometa. Nadalje, bitno je da generator prometa koristi razne protokole kako bi se otežalo detektiranje generiranog prometa. Tablica 3.2 prikazuje različite protokole koje je alat Noisy generirao prilikom rada. Sav promet koji je generiran je dio IPv4 protokola. UDP protokolu pripada 6,6% podataka, DHCP protokolu pripada mali broj paketa, a većina UDP paketa pripada DNS zahtjevima koji

su generirani prilikom pretraživanja stranica. Ostatak paketa (93,4%) pripada TCP protokolu gdje je većina paketa pripada TLS protokolu koji je zadužen za sigurnu komunikaciju.

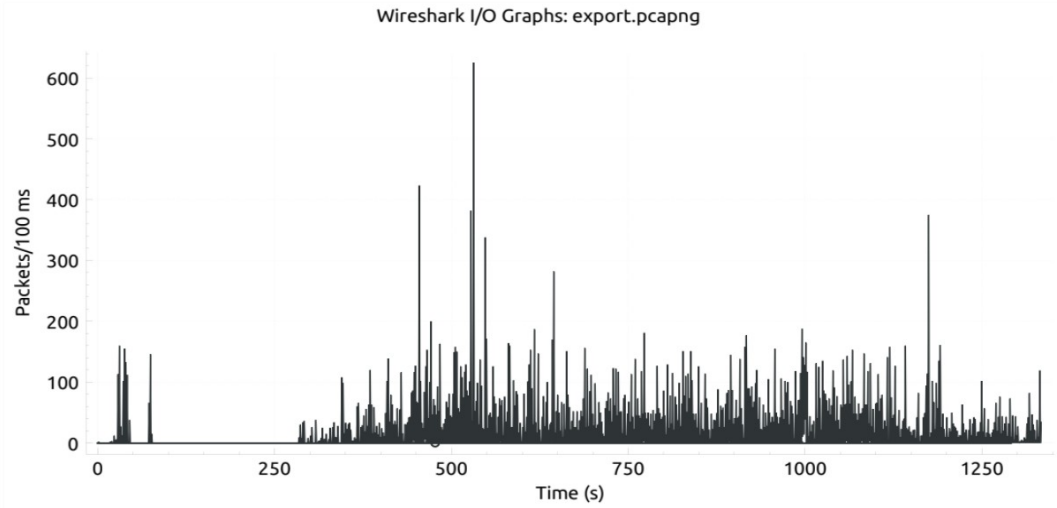
Protokol	Postotak paketa koji pripada protokolu	Broj paketa	Postotak okteta od svih okteta koje je protokol generirao	Broj okteta	Bit/s
Okvir	100	141825	100,00	101445744	608796,76
Ethernet	100	141825	1,96	1985550	11915,69
Internet Protocol Version 4	99,954	141760	2,79	2835200	17014,62
User Datagram Protocol	6,600	9360	0,07	74880	449,37
Dynamic Host Configuration Protocol	0,035	49	0,02	20652	123,94
Domain Name System	6,565	9311	0,52	526527	3159,80
Transmission Control Protocol	93,354	132400	94,57	95935517	575728,75
Transport Layer Security	35,597	50486	94,46	95824734	575063,92
Hypertext Transfer Protocol	0,488	692	0,82	829150	4975,90
Online Certificate Status Protocol	0,023	32	0,01	8868	53,22
Line-based text data	0,058	82	2,06	2086262	12520,09
eXtensible Markup Language	0,001	1	0,20	200096	1200,82
Compuserve GIF	0,001	1	0,00	43	0,26
Address Resolution Protocol	0,046	65	0,00	2288	13,73

Tablica 3.2 Korišteni protokoli generiranog prometa

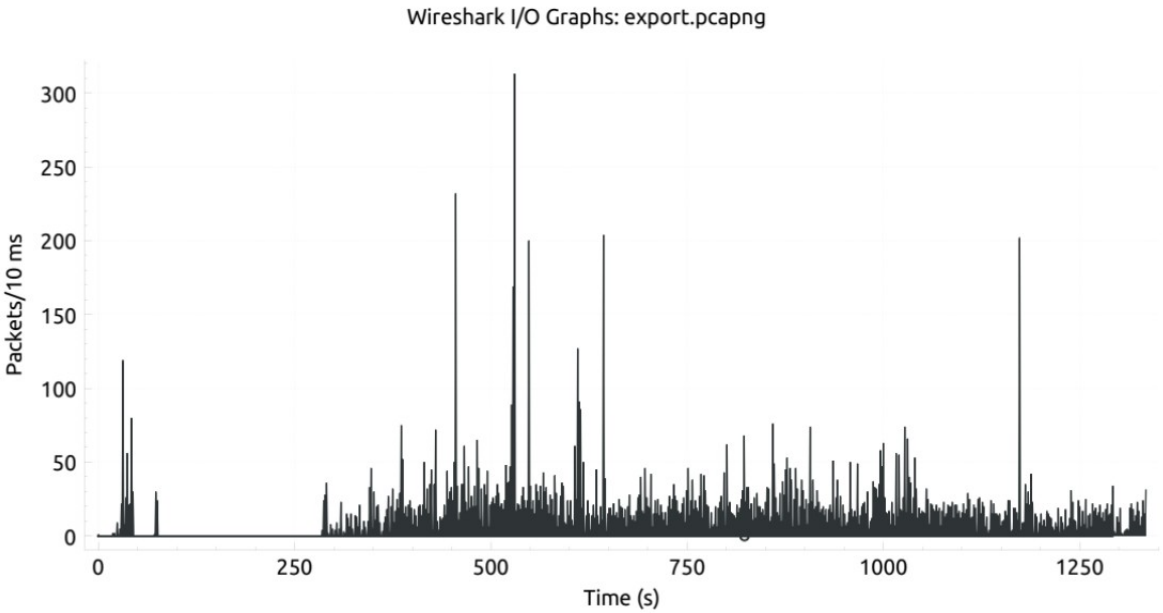
Slike 3.4, 3.5. i 3.6 prikazuju grafove koji su dobiveni prikazivanjem broja paketa poslanih i primljenih po jedinici vremena. Svaka od slika prikazuje različite intervale.



Slika 3.4 Graf generiranog prometa s intervalom: 1 sekunda

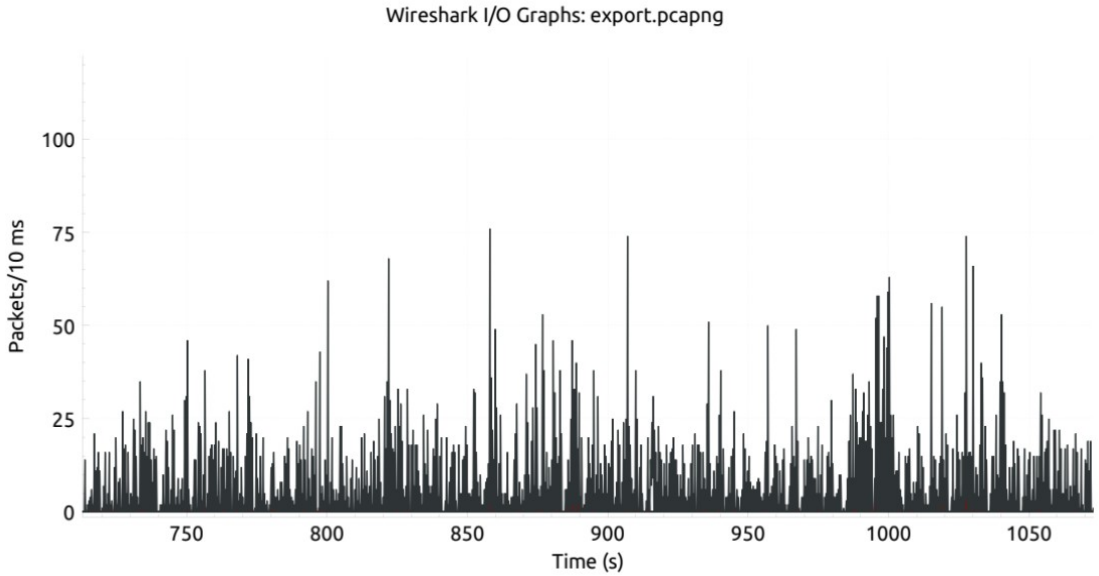


Slika 3.5 Graf generiranog prometa s intervalom: 100 ms



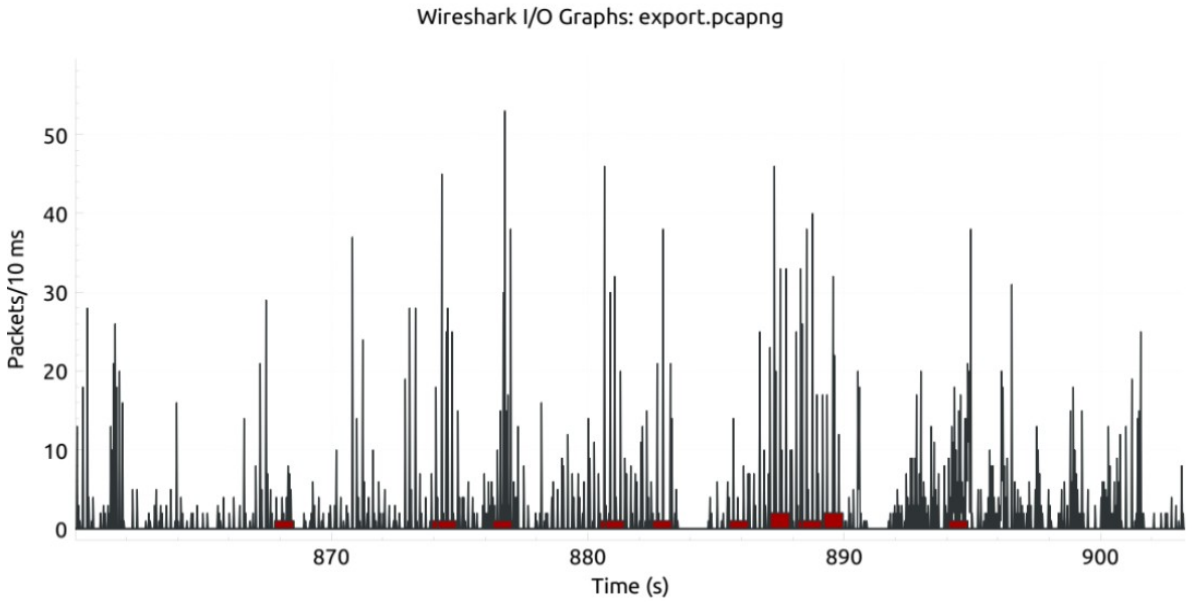
Slika 3.6 Graf generiranog prometa s intervalom: 10 ms

Zanimljivo je kako se na grafovima može primijetiti da podsjećaju na fraktale. Naime uvećavanjem prikaza grafa na slici 3.6 dobiva se slika 3.7



Slika 3.7 Graf generiranog prometa s intervalom: 10 ms, uvećani prikaz između 700s i 1100s

Daljnjim uvećavanjem grafa sa slike 3.7 dobiva se graf prikazan na slici 3.8.



Slika 3.8 Graf generiranog prometa s intervalom: 10 ms, uvećani prikaz između 860s i 905s

Grafovi se doimaju sličnima. To je svojstvo koje imaju fraktali i brojne druge stvari koje u prirodi koje prate fraktalni uzorak. Daljnjim povećavanjem grafova dobivali bi slične prikaze. To je zato što generirani podaci imaju periode kada se generira velik broj podataka i zatim periode kada se ne generiraju podaci, zato se ovi grafovi doimaju skokovitima.

3.4. Fraktalnost mreže

Promet koji ima brojne "skokove" u količini odaslanih paketa u nekom vremenskom intervalu se statistički može opisati pomoću samo-sličnosti podataka. Promet koji je generiran alatom Noisy je skokovit i nalikuje na fraktalnu sliku, prikazano je u grafovima 3.4, 3.5, 3.6, 3.7 i 3.8. Samo sličnost je svojstvo koje posjeduju fraktali, a to znači da podaci izgledaju jednako bez obzira na vremenski interval na kojem ih promatramo. Kako bi se dokazala samo-sličnost prometa koje je proizveo generator prometa potrebno je provesti analizu nad podacima.

Samo-slični [19] skup vremena ima svojstvo da mu nalikuju njegovi podskupovi. Neka je $X=(X_t: t=0,1,2,\ldots)$ stacionarna kovarijanca stohastičkog procesa. To je zapravo proces s konstantnom srednjom vrijednosti $\mu=E[X_t]$, konačnom varijancom $\sigma^2=E[(X_t-\mu)^2]$ i autokorelacijskom funkcijom $r(k)=E[(X_t-\mu)(X_{t+k}-\mu)]/E[(X_t-\mu)^2](k=0,1,2,\ldots)$ koja ovisi samo o varijabli k. Pretpostavljamo da X ima autokorelacijsku funkciju oblika $r(k)\sim a_1 k^{-\beta}, kada\, k\rightarrow\infty$ gdje je $0<\beta<1$, a $a_1, a_2, \ldots$ označava konačan skup pozitivnih konstanti.

Za svaki $m=1,2,3,\ldots$ neka je $X^m=1/m(X_{km-m+1}+\ldots+X_{km}),(k\geqslant 1)$.

Za svaki m, agregirani skup vremena X^m definira stacionarnu kovarijancu procesa. Proces X je samo sličan s parametrom $H=1-\beta/2$ gdje H označava parametar samosličnost, ako procesi X i X^m koreliraju. Vidljivo je na grafovima koji prikazuju broj paketa u vremenu da neovisno o veličini intervala da grafovi izgledaju slično, što intuitivno upućuje da postoji korelacija između tih grafova.

3.4.1. Hurstovo pravilo

Samo-slični procesi su primjer empirijskog zakona koji se naziva Hurstovo pravilo ili Hurstov efekt [20]. Pravilo je objavio hidrolog H.E. Hurst u radu naziva "The Long-Term Storage Capacity of Reservoirs" 1951. godine. Iako se rad bavio isključivo modeliranjem rezervoara na rijeci Nil, pokazalo se da je rad primjenjiv u brojnim drugim znanstvenim granama. Nakon puno godina zakon je dobio na popularnosti kad je Benoit Mandelbrot uočio potencijal u radu te ga iskoristio u fraktalnoj geometriji. Hurstov eksponente koristi kako bi se mjerila dugoročna memorija vremenskog niza. Hurstov eksponent H je definiran sljedećim izrazom:

$$E[\frac{R(n)}{S(n)}]=Cn^H, dok\ n\rightarrow\infty$$

pri čemu je:

$R(n)$ je skup prvih n kumulativnih devijacija od srednje vrijednosti
$S(n)$ je standardna devijacija
$E[x]$ je očekivana vrijednost
n je vremensko trajanje promatranja tj. broj podataka unutar vremenskog skupa podataka
C je konstanta.

Hurstov eksponent poprima vrijednosti između 0 i 1. Ako je dobiveni eksponent između 0,5 i 1 tada to upućuje da vremenski skup podataka ima dugotrajnu korelaciju. Hurstov eksponent za samo-slične skupove podataka bi trebao poprimiti upravo tu vrijednost između 0,5 i 1. Ako je dobivena vrijednost između 0 i 0,5 to upućuje da je će vremenski skup nakon niskih vrijednosti skočiti i poprimiti visoke vrijednosti. Takvi skupovi podataka na vremenskom grafu izgledaju skokovito. Ako je vrijednost eksponenta jednaka 0.5 tada to znači da ne postoji korelacija.

3.4.2. Hurstov eksponent generiranog prometa

S ciljem da se prouče svojstva generiranog prometa s alatom Noisy, potrebno je bilo snimiti promet koji je generiran na sučelju i zatim odrediti Hurstov eksponent za dobivene podatke. Povod tome je bilo istraživanje u radu "On the Self-Similar Nature of Ethernet Traffic" [19]. U radu je opisan eksperiment gdje je sniman sav Ethernet promet na sveučilištu Bellcore. Promet je sniman kroz 4 godine kako bi se dobili što raznovrsniji podaci. Pogodno je za usporedbu s Noisy alatom zato što je ovaj promet generiran od strane stvarnih korisnika, dok je Noisy "imitirao" korisnika koji pretražuje po internetu. U radu je dokazano da snimljeni podaci imaju Hurtov eksponent približno 0.8 što povlači samo-sličnost prometa.

Podatke koje koji su generirani na topologiji i snimljeni alatom Wireshark, prebačeni su u *csv* format kako bi bili pogodniji za daljnju obradu. Radi lakšeg izračunavanja Hurstovog eksponenta iskorištena je Python biblioteka "hurst" [23]. Biblioteka implementira funkciju „calculate_Hc” koja računa Hurstov eksponent. Prvi argument funkcije je lista koja sadrži

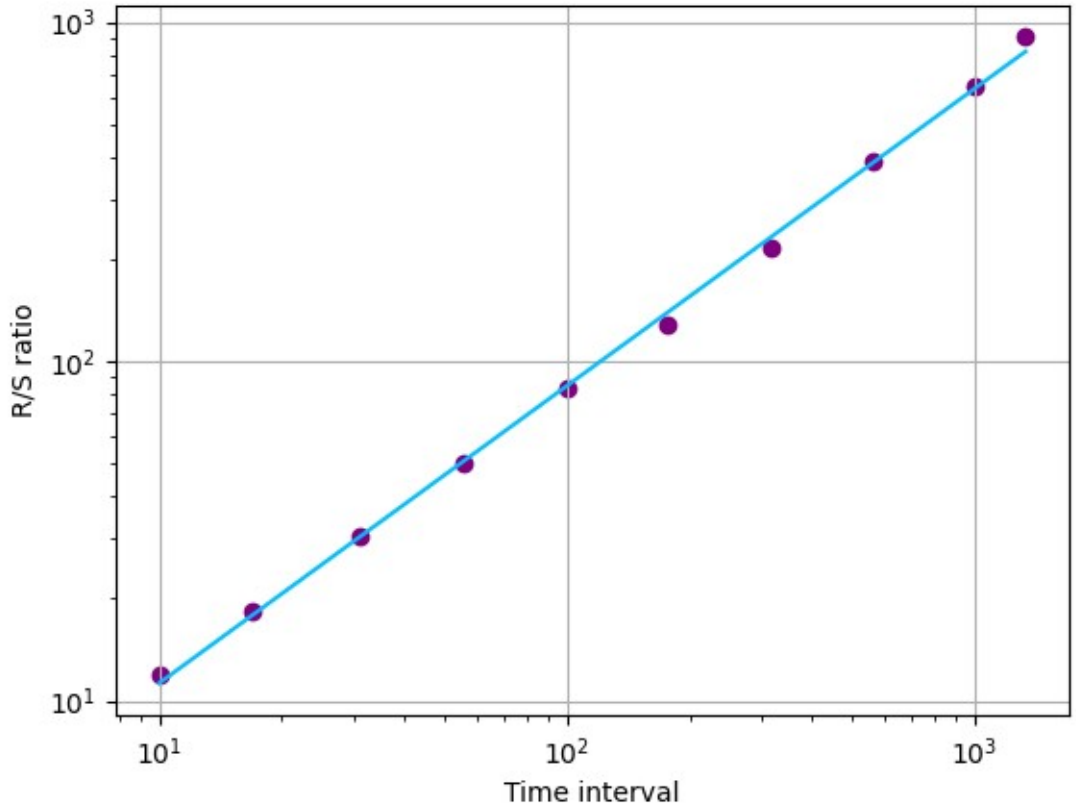
inkrementiran vremenski niz što znači da svakom članu liste dodijelimo vremenski interval kojeg opisuje i u članu liste je zapisan broj podataka koji su generirani u tom intervalu. Sljedeći argument je vrsta skupa podataka. Razlikuju se tri moguće vrijednosti, 'change' kao skup nasumičnih vrijednosti, 'random_walk' kao skup kumulativnih suma promjena, 'price' kao skup kumulativnih produkata promjena. Posljednji argument funkcije služi da bi se postavilo koji izraz koristiti, pojednostavljenu ili originalnu R/S kalkulaciju.

Python skripta prikazana u Ispisu 3.2 učitava *csv* datoteku i sve zapise vremena sprema u listu podataka. Nakon toga stvara listu veličine broj sekundi × rezolucija. Rezolucija je u kodu prikazana varijablom N. Što je veća varijabla N bit će i više vremenskih podataka u listi. Svakom indeksu liste se pridodaje vrijednost koja odgovara broju podatka koji su generirani unutar vremenskog intervala. Takva lista inkrementa se predaje funkciji 'compute_Hc' koja izračunava Hurstov eksponent i iscrtava R/S graf. Program kao ulazne argumente prima ime datoteke koja odgovara izvezenoj datoteci iz alata Wireshark, maksimalnu vrijednost sekunde u *csv* datoteci te broj koji predstavlja vrijednost 1/N. Za primjer koji je snimljen na prethodno spomenutoj topologiji pokreće se: *packets_per_sec.py export_big.csv 1334 1*

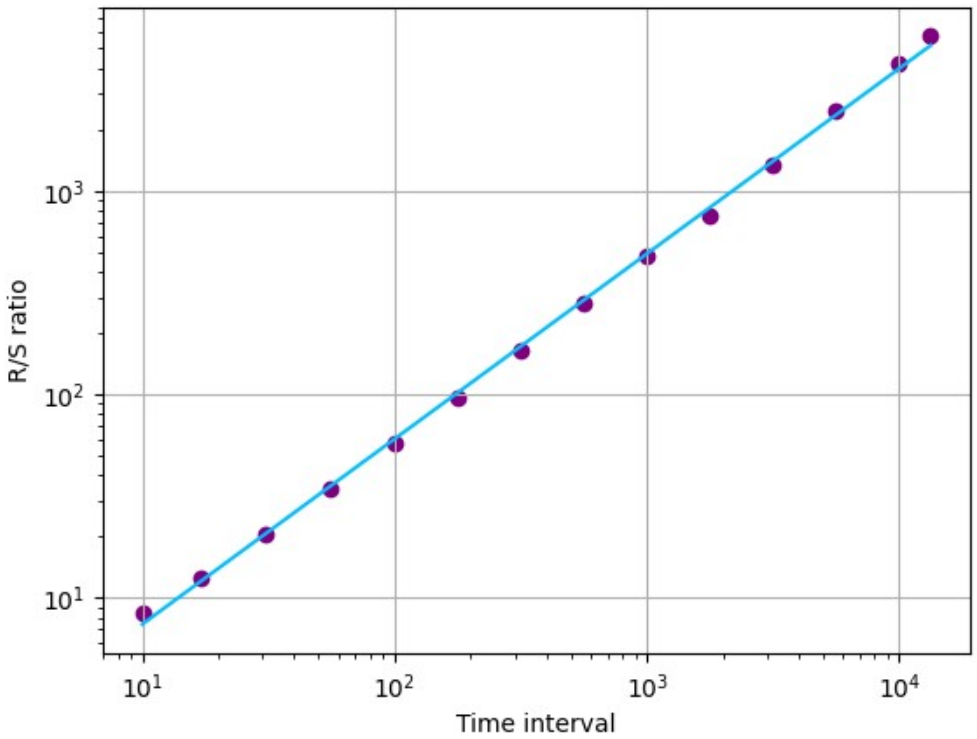
```
import numpy as np
import matplotlib.pyplot as plt
from hurst import compute_Hc
import csv
import sys

def split_by_n(list_input, N):
    list_tmp = [0] * round(seconds * N)
    for time in list_input:
        list_tmp[round(time * N)] += 1
    return list_tmp

if __name__ == '__main__':
    filename = sys.argv[1]
    seconds = int(sys.argv[2])
    N = int(sys.argv[3])
    with open(filename) as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=',')
        next(csv_reader)
        list = []
        for row in csv_reader:
            list.append(float(row[1]))

    list_timeseries = np.array(split_by_n(list, N))
    H, c, data = compute_Hc(list_timeseries, kind='change', simplified=True)
    f, ax = plt.subplots()
    ax.plot(data[0], c * data[0] ** H, color="deepskyblue")
    ax.scatter(data[0], data[1], color="purple")
    ax.set_xscale('log')
    ax.set_yscale('log')
    ax.set_xlabel('Time interval')
    ax.set_ylabel('R/S ratio')
    ax.grid(True)
    plt.show()
```

Ispis 3.2 Python skipta kojom su rađeni grafovi i računati eksponenti

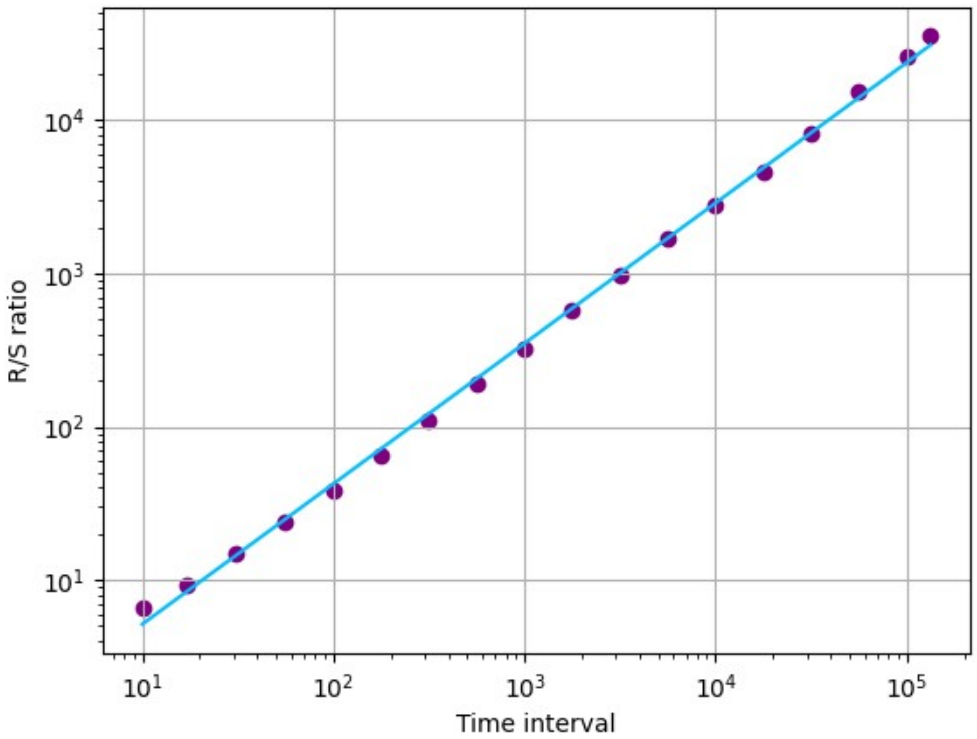
Dobiveni R/S grafovi s izračunatim Hurstovim eksponentom prikazani su na slikama 3.9, 3.10 i 3.11. Grafovi se međusobno razlikuju u argumentu N koji odlučuje o rezoluciji. Što je rezolucija veća, veći je i broj točaka kojima se računa koeficijent. Razlike između Hurstovih koeficijenata pojedinih grafova su male. Povećavanjem rezolucije očekivano je da sličnost bude veća što se podudara s dobivenim rezultatima. U radu "On the Self-Similar Nature of Ethernet Traffic" dobiveni Hurstov eksponent je približno jednak 0.80. Činjenica da je Hurstov eksponent prometa generiranog alatom Noisy veći nego u navedenom radu je i za očekivati zato što su promet u radu generirali stvarni korisnici računala, a ne alat. Iako je Hurstov eksponent generiran alatom približno jednak 0.9 to je i dalje obećavajući rezultat.



Slika 3.9 R/S graf s vrijednostima N = 1, H=0.8757



Slika 3.10 R/S Graf s vrijednostima N = 10, H=0.9098



Slika 3.11 R/S graf s vrijednostima N = 100, H=0.9163

4. Zaključak

U radu je objašnjeno na koji način se organiziraju kibernetički poligoni i koje su ključne komponente kibernetičkog poligona. Organizacija kibernetičkog poligona se ostvaruje raspoređivanjem korisnika vježbe u različite timove. Svaki od timova ima posebna zaduženja tijekom vježbe. Arhitektura kibernetičkog poligona je podijeljena u četiri nužne komponente, virtualnu komponentu koja oponaša arhitekturu stvarnog sustava, komponentu koja omogućuje izvođenje raznih vrsta napada, komponentu koja generira korisnički promet i komponentu koja služi praćenju stanja unutar virtualnog okruženja. U radu su navedeni različiti primjeri kibernetičkih poligona.

Nakon toga se proučavaju različiti alati koji bi se mogli iskoristiti kao generatori prometa. Svaki od opisanih alata također ima opisane nedostatke. Alat Noisy se u teoriji pokazao alatom kojim bi se mogao ostvariti generator prometa te je zato intenzivnije objašnjen i urađene su preinake u datoteci s početnim postavkama.

Kako bi se dokazalo da je alat Noisy uistinu zadovoljavajuće rješenje, potrebno je napraviti simulaciju u kojoj će se iskoristiti alat i proučiti sintetički generirani promet. Kao rješenje okoline u kojoj će se izvoditi simulacija korišten je alat Imunes i opisani su svi nužni koraci kako bi se odgovarajuća okolina postavila.

Stvarne mreže s pravim korisnicima imaju svojstvo samo-sličnosti koje je detaljnije pojašnjeno. Za potrebe dokaza samo-sličnosti generiranih podataka napisana je kratka skripta u programskom jeziku Python koja računa koeficijent samo-sličnosti, zvan Hurstov eksponent. Pokazalo se da podaci uistinu nalikuju stvarnim podacima i to na par načina. Prvo koriste se različiti protokoli koje i stvarni korisnici računala koriste, podaci se generiraju u nepredvidivim vremenskim intervalima i Hurstov eksponent dokazuje da je mreža samo slična te je koeficijent približno jednak kao koeficijent dobiven u radu "On the Self-Similar Nature of Ethernet Traffic".

Potrebno je spomenuti da postoje još moguća poboljšanja. Prvenstveno bi trebalo vremena slanja paketa više učiniti nepredvidivima. Također nagli skokovi su u ovom eksperimentu mogući su radi male propusnosti mreže i velikog broja računala koja traže pretražuju podatke internetom. Uklanjanjem problema propusnosti bi se dobili znatno manje samo-slični podaci. Korisno bi također bilo kada bi u mreži bilo biše računala koja bi generirala promet, ali da komuniciraju s vanjskim internetom preko više različitih sučelja. Podaci bi bili kvalitetniji kada bi jedno računalo generiralo zahtjev npr. svakih 20 sekundi i kada bi se simulacija pustila da generira podatke duže vrijeme. Unaprjeđenje rada bi bilo kada bi se uz Noisy integrirala i druga potencijalna rješenja kao DNoiSe ili još neki drugi alat koji bi generirali samostalno podatke. Nedostatak alata Noisy je da pretraživanjem interneta ne koristi puno protokola, iako koristi većinu protokola koje korisnik koristi.

5. Literatura

[1] Varonis (2020. godine) Data Breach Response Times: Trends and Tips https://www.varonis.com/blog/data-breach-response-times (pristupljeno 31. svibnja 2020.)

[2] Accenture (2019. godine) Ninth Annual Cost of Cybercrime Study https://www.accenture.com/us-en/insights/security/cost-cybercrime-study (pristupljeno 31. svibnja 2020.)

[3] Verizon (2020. godine) 2020 Data Breach Investigations Report https://enterprise.verizon.com/en-gb/resources/reports/dbir (pristupljeno 31. svibnja 2020.)

[4] Cloudshare (2020. godine) What Is a Cyber Range?, https://www.cloudshare.com/virtual-it-labs-glossary/what-is-a-cyber-range (posjećeno 31. svibnja 2020.)

[5] ScienceDirect (2020. godine) Cyber ranges and security testbeds: Scenarios, functions, tools and architecture https://www.sciencedirect.com/science/article/pii/S0167404819301804 (posjećeno 31. svibnja 2020.)

[6] Cyberbit (2019. godine) Cyber Security Training Platform https://www.cyberbit.com/blog/cybersecurity-training/cyber-security-training-platform/ (posjećeno 31. svibnja 2020.)

[7] OCCP (2020), OCCP dokumentacija, https://opencyberchallenge.net/ (posjećeno 31. svibnja 2020.)

[8] Github (2020),CyberRange dokumentacija, https://github.com/secdevops-cuse/CyberRange (posjećeno 31. svibnja 2020.)

[9] Cyberbit letak, Cyber Range Buyers Guide for Security Service Providers https://www.infosecurityeurope.com/__novadocuments/467451?v=636590390539170000 (posjećeno 31. svibnja 2020.)

[10] Cisco Cyber Range prezentacija, https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwitzvPk7YHqAhXDlIsKHfp4CXIQFjACegQIBBAB&url=https%3A%2F%2Fwww.ciscolive.com%2Fc%2Fdam%2Fr%2Fciscolive%2Fapjc%2Fdocs%2F2016%2Fpdf%2FBRKSEC-2653.pdf&usg=AOvVaw2BiZhjnCIhbFGrM4tSA-LM (preuzeto 31.svibnja 2020.)

[11] Ostinato (2020. godine) Packet Generator https://ostinato.org/ (posjećeno 31. svibnja 2020.)

[12] Github (2020), DNoiSe dokumentacija, https://github.com/jankais3r/DNoiSe (posjećeno 31.svibnja 2020.)

[13] Cisco Umbrella 1 Million, referenca na listu domena, https://umbrella.cisco.com/blog/cisco-umbrella-1-million (posjećeno 31. svibnja 2020.)

[14] D-ITG, dokumentacija alata, http://www.grid.unina.it/software/ITG/ (posjećeno 31. svibnja 2020.)

[15] Github (2018. godine) Noisy dokumentacija alata https://github.com/1tayH/noisy (posjećeno 31. svibnja 2020.)

[16] Github (2020. godine) web-traffic-generator dokumentacija alata https://github.com/ReconInfoSec/web-traffic-generator (posjećeno 31. svibnja 2020.)

[17] Wikipedia (2020. godine) Web crawler https://en.wikipedia.org/wiki/Web_crawler (posjećeno 31. svibnja 2020.)

[18] Službene upute za korištenje Imunes alata http://imunes.net/dl/imunes_user_guide.pdf (dohvaćeno 31. svibnja 2020.)

[19] Will E. Leland, Walter Willinger, Murad S. Taqqu, Daniel V. Wilson, (1993. godine), On the Self-Similar Nature of Ethernet Traffic, https://www.cs.auckland.ac.nz/courses/compsci742s2c/resources/LTWW93.pdf (dohvaćeno 31. svibnja 2020.)

[20] The Hurst Exponent: Predictability of Time Series, (2012. godine), The Hurst Exponent: Predictability of Time Series, http://analytics-magazine.org/the-hurst-exponent-predictability-of-time-series/ (posjećeno 31. svibnja 2020.)

[21] Wikipedia, Hurst exponent, https://en.wikipedia.org/wiki/Hurst_exponent (posjećeno 31. svibnja 2020.)

[22] Mark E. Crovella, Azer Bestavros, (1995. godine) Explaining World Wide Web Traffic Self-Similarity, http://www.cs.bu.edu/fac/crovella/paper-archive/self-sim/paper.html (posjećeno 31. svibnja 2020.)

[23] Github, dokumentacija Hurst python biblioteke, https://github.com/Mottl/hurst (posjećeno 31. svibnja 2020.)

[24] AWS, službena stranica ,https://aws.amazon.com/ (posjećeno 31. svibnja 2020.)

[25] Kali Linux službena stranica, https://www.kali.org/ (posjećeno 31. svibnja 2020.)

[26] Nessus službena stranica https://www.tenable.com/products/nessus (posjećeno 31. svibnja 2020.)

[27] Github, Command-VM dokumentacija, https://github.com/fireeye/commando-vm (posjećeno 31. svibnja 2020.)

[28] Terraform službena stranica, https://www.terraform.io/ (posjećeno 31. svibnja 2020.)

[29] Docker službena stranica, https://www.docker.com/ (posjećeno 31. svibnja 2020.)

[30] Github, DetectionLab službena dokumentacija, https://github.com/clong/DetectionLab (posjećeno 31. svibnja 2020.)

[31] Inspec službena dokumentacija, https://www.inspec.io/docs/ (posjećeno 31. svibnja 2020.)

[32] Dev.to definicija CTF-a, https://dev.to/atan/what-is-ctf-and-how-to-get-started-3f04 (posjećeno 31. svibnja 2020.)

[33] IxNetwork službena stranica, https://www.ixiacom.com/products/ixnetwork (posjećeno 31. svibnja 2020.)