



Univerzitet u Beogradu
Elektrotehnički fakultet

DIZAJN I IMPLEMENTACIJA INTERAKTIVNE APLIKACIJE ZA VIZUELNI PRIKAZ DDOS NAPADA

DIPLOMSKI RAD

Kandidat:
Aleksa Novaković
2017/0305

Profesor:
dr Žarko Stanisavljević,
vanredni profesor

Beograd
septembar 2021.

Sadržaj

1. UVOD.....	3
2. OPIS PROBLEMA	4
2.1. PREGLED POSTOJEĆIH REŠENJA.....	4
2.2. PREDLOG REŠENJA.....	5
2.3. DOS I DDOS	5
2.4. KONCEPT BOTNET MREŽA	8
2.5. TIPOVI DDOS NAPADA	9
3. IMPLEMENTACIJA APLIKACIJE.....	14
3.1. RAZVOJNO I IMPLEMENTACIONO OKRUŽENJE	14
3.2. DIZAJN APLIKACIJE	14
3.3. IMPLEMENTACIJA FORMI U OKVIRU APLIKACIJE	18
3.4. IMPLEMENTACIJA SAJTOVA	19
3.5. IMPLEMENTACIJA DDOS NAPADA	21
3.6. OPIS KORIŠĆENIH KLASA HOSTOVA.....	23
4. NAČIN KORIŠĆENJA APLIKACIJE	30
4.1. PREDUSLOV ZA POKRETANJE APLIKACIJE	30
4.2. OSNOVNI PROZOR APLIKACIJE I GENERISANJE MREŽE	30
4.3. DDOS NAPADI, POKRETANJE I ZAUSTAVLJANJE NAPADA	34
4.4. PRIMERI DDOS NAPADA	36
5. ZAKLJUČAK.....	39
LITERATURA.....	40
PRILOG A.....	43
OSI REFERENTNI MODEL	43
MIRAI BOTNET	45
DDOS KAO SERVIS	45

1. UVOD

Nastanak interneta, kao najvećeg distribuiranog računarskog sistema i mrežnih protokola, imali su veliki uticaj na svakodnevni život i razvoj nauke poslednjih nekoliko decenija. Danas, gotovo sve velike kompanije se oslanjaju na internet u velikoj meri za svoje funkcionisanje i za komunikaciju sa svojim klijentima. Sve veći broj institucija se oslanja na internet radi efikasnijeg i lakšeg poslovanja.

Takođe, razvojem kriptografije, internet je omogućio postojanje virtualnih valuta koje koriste koncept tzv. *blockchain-a*, kriptografskog sistema koji u svojoj osnovi zahteva postojanje interneta kao medijuma za prenos informacija između računara koji učestvuju u formiranju *blockchain-a* i obradi transakcija.

Gotovo je nemoguće navesti sve primene i slučajeve korišćenja interneta. Dakle, internet je postao kritična infrastruktura u funkcionisanju mnogih servisa i neometan pristup njemu je izuzetno bitan, a problemi u njegovom funkcionisanju mogu da dovedu do velikih posledica koje mogu da ugroze nečije poslovanje, imovinu, čak i da ugroze živote ljudi.

Iz aspekta računarske bezbednosti, kod svakog sistema su bitna tri glavna elementa: poverljivost, integritet i dostupnost. [1]

Jednu od glavnih pretnji na dostupnost servisa na internetu svakako predstavljaju *Denial of Service (DoS)* i *Distributed Denial of Service (DDoS)* napadi. Ove vrste napada su postojale od nastanka računarskih mreža i mrežnih protokola i pratili su njihov razvoj, postavši sofisticirani i razorniji.

U okviru studijskog programa računarske tehnike i informatike, sa različitim tipovima mrežnih napada studenti se susreću na predmetima Računarske mreže 1, Računarske mreže 2, dok je koncept *DoS* i *DDoS* napada posebno obrađen na predmetu Zaštita podataka. Naravno, vrlo je bitno razumevanje *DDoS* napada kao koncepta i u cilju toga je nastala ideja o aplikaciji koja bi vizuelno prikazala i dodatno studentima objasnila pomenuti koncept.

U drugom poglavlju rada su izložena postojeća rešenja za vizuelni prikaz *DDoS* napada, dat je predlog novog rešenja, kao i detaljnija analiza problema *DDoS* napada. U trećem poglavlju dat je detaljan opis implementacije rešenja, okruženja i platforme koja se koristi, prikazan je dizajn formi i implementacija različitih tipova *DDoS* napada. U okviru četvrtog poglavlja prikazan je način korišćenja aplikacije, zajedno sa primerima koji ilustruju efekte *DDoS* napada. U petom poglavlju se daje zaključak o *DDoS* napadima, kao i o implementaciji aplikacije i mogućim poboljšanjima. U okviru priloga dat je pregled *OSI* referentnog modela, *DDoS-for-hire* sistema kao i *Mirai botnet-a*.

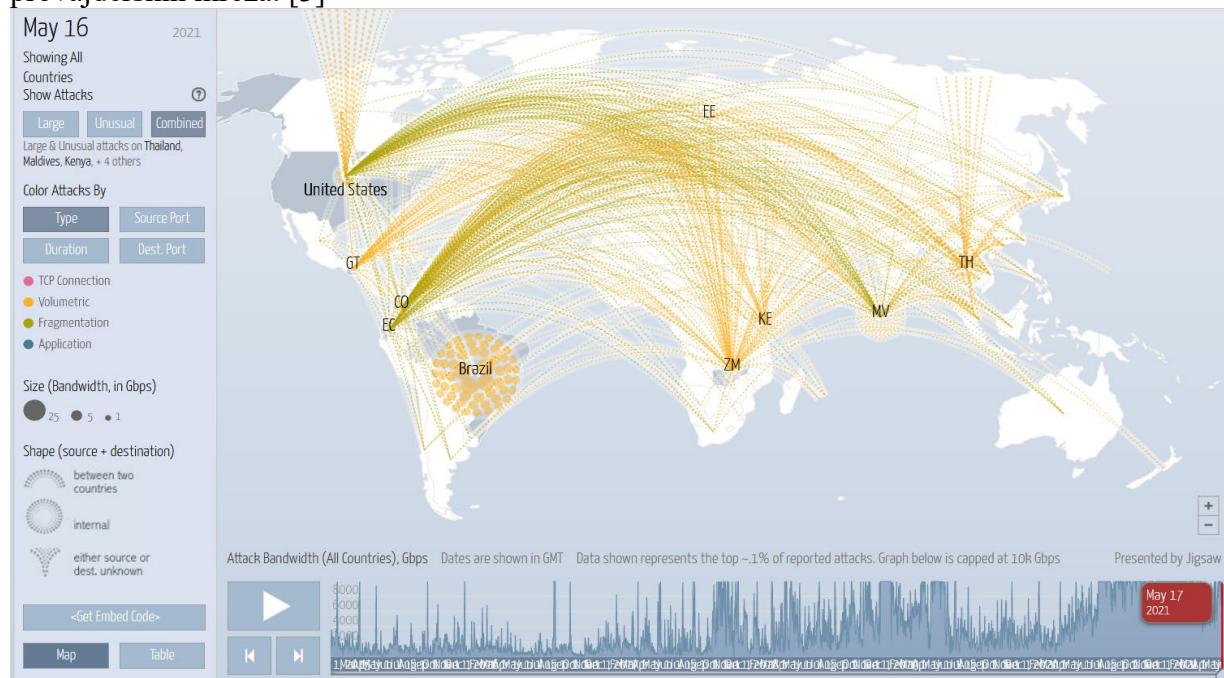
2. OPIS PROBLEMA

U ovom poglavlju je izložen pregled postojećih rešenja, opisan predlog rešenja i data detaljnija analiza problema za koji se rešenje realizuje, kroz sagledavanje koncepta mrežnih protokola, *botnet* mreža i razlikama između klasifikacije *DoS* i *DDoS* napada, čije je razumevanje vrlo bitno da bi se pravilno razumeo rad aplikacije rešenja.

2.1. PREGLED POSTOJEĆIH REŠENJA

Postoje određene *web* aplikacije koje obavljaju funkciju vizuelnog prikaza *DDoS* napada, međutim one nemaju edukativan karakter i ne mogu da na blizak način prikažu šta se stvarno događa.

Jednu od interesantnijih *web* aplikacija za prikaz *DDoS* napada predstavljaju digitalne mape, gde su na mapi sveta, podeljenoj na države, grafički prikazani napadi kao linije različite debljine (u odnosu na jačinu napada), koje počinju u državi odakle dolazi napad a završavaju na meti napada. Takođe je moguće videti i tip napada. Jedna od tih mapa je *Digital Attack Map* koju su razvile kompanije *Arbor Networks* i *Google Ideas*. Podaci koji su prikazani na mapi su prikupljeni na osnovu inteligentne mreže senzora koji se nalaze u preko 330 provajderskih mreža. [3]



Slika 2.1 – Prikaz mape Digital Attack Map za datum 16.5.2021. god. [3]

Slika 2.1 sadrži prikaz detektovanih napada za 16. maj 2021. godine. Nažalost izgleda da ova *web* aplikacija nije više funkcionalna, budući da ne sadrži prikaze napada koji su se dogodili nakon navedenog datuma.

2.2. PREDLOG REŠENJA

Za formiranje početne ideje o aplikaciji poslužile su implementacije simulacionih programa sa kojima smo se susreli na Računarskim mrežama 1 i 2 – *GNS3* [25] i *Cisco Packet Tracer* [26], koje su studentima omogućili da na vrlo jednostavan i interesantan način shvate funkcionisanje i konfiguraciju mrežnih uređaja i rad mrežnih protokola. U ovim okruženjima student ima opciju da instancira mrežne uređaje, povezuje ih linkovima, da vrši konfiguraciju mrežnih protokola u okviru terminala svakog uređaja i da vrši proveru funkcionisanja protokola i povezanosti korišćenjem *ICMP* [27] ili *HTTP* [28] protokola u okviru pregledača i komandne linije.

Naravno, budući da za korišćenje aplikacije ne bi trebalo da bude neophodno znanje za konfiguraciju mreže, aplikacija bi za korisnika izgenerisala već povezanu i konfigurisanu mrežu, na koju će on moći da utiče sa osnovnim konfiguracionim parametrima (broj *host*-ova, mrežni kapaciteti...).

U aplikaciji korisnik bi mogao da posmatra celu mrežu, kao u simulacionim okruženjima, da ima mogućnost pomeranja i vizuelnog preuređenja *host*-ova, a pritom da vizuelni prikaz ne bude zakomplikovan prikazom velikog broja *host*-ova istovremeno. Na radnoj površini bi takođe bili prikazani detalji svakog selektovanog *host*-a – tip uređaja, kapacitet mrežnog linka na koji je povezan, *delay* do *target* servera i njegova *IPv4* adresa.

Naravno, u okviru aplikacije korisnik bi imao mogućnost da započne jedan od više vrsta *DDoS* napada iz kontrolnog panela za *botnet*, efektivno simulirajući ulogu tzv. *botmaster*-a, selektovanjem određenog broja botova koji će da izvrše napad, da ih zaustavi i da vidi kakav efekat to ima na metu napada i na celu mrežu – kako se menjaju trenutne iskorišćenosti linkova itd.

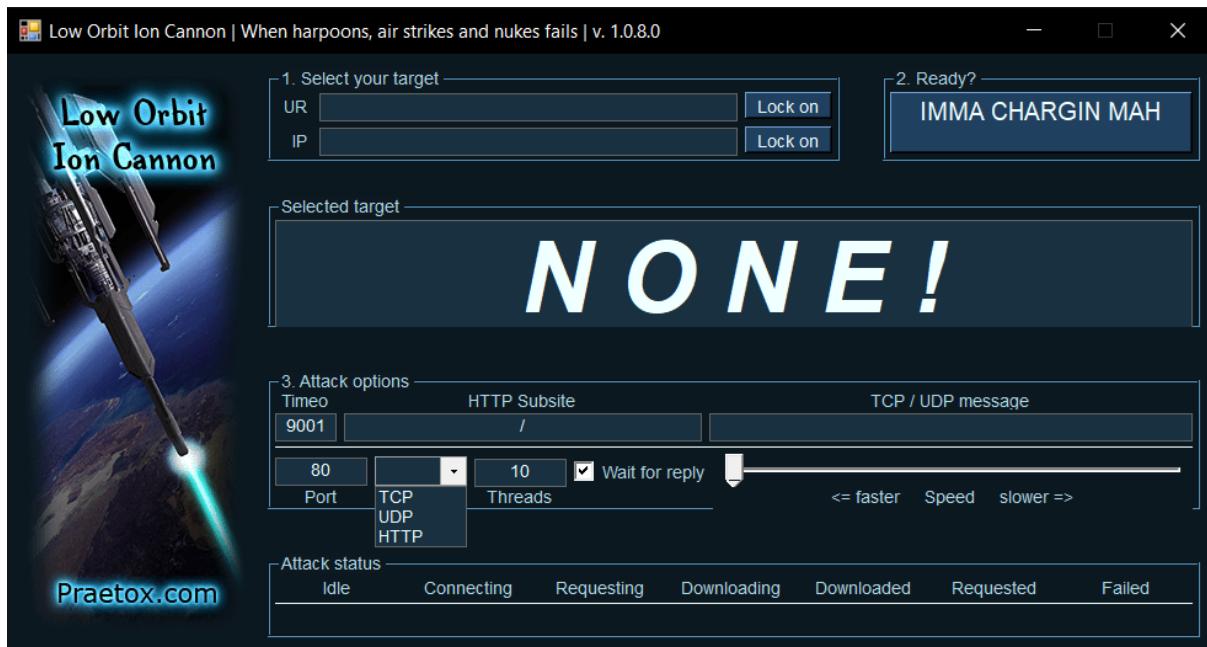
Meta napada bi bila implementirana kao jednostavan *HTTP* sajt, budući da su sajtovi statistički i najčešće mete ovakvih napada, a korisnik bi sajtu mogao da pristupi preko ugrađenog *web* pregledača, kucanjem određene *URL* adrese, koja se interno razrešava u okviru aplikacije u odgovarajući sajt, ukoliko takav sajt postoji i ukoliko je moguće da mu se pristupi.

Dodatna ideja je i da aplikacija ima implementiranu i funkciju logovanja napada koja bi omogućila korisniku da ima uvid u mrežu iz ugla mrežnog administratora – može da vidi kada su napadi započeli, kada su se zaustavili, odakle dolaze i kome su namenjeni napadi, kao i tip samog napada.

2.3. DOS I DDOS

DoS (Denial of Service) [1] je napad koji koristi internet konekciju i zloupotrebljava određeni mrežni protokol radi onemogućavanja legitimnim korisnicima da pristupe servisu koji je meta napada ili radi degradiranja performansi funkcionisanja napadnutog sistema.

Napad je *DoS* ako on dolazi od samo jednog napadača, tjst. jedne mašine i on je zbog toga lakši za detektovanje i za blokiranje. Tokom godina pojavile su se mnoge besplatne i *open-source* aplikacije koje služe za pokretanje ovakvih napada - *LOIC* (*Low Orbit Ion Cannon*, prikazana na slici 2.2) [29], *HOIC* (*High Orbit Ion Cannon*) [30] ili *Slowloris* [31] koje omogućavaju ljudima bez tehničkog znanja da pokrenu napade i u velikim brojevima i ovi napadi mogu biti pogubni po metu.



Slika 2.2 – Korisnički interfejs LOIC programa za DoS napade

DDoS (Distributed Denial of Service) [1] napadi predstavljaju koordinisane napade veće razmere, koji imaju mnogo veći uticaj na mrežne ili sistemske resurse mete, u odnosu na *DoS* napade. Ovakvi napadi se pokreću sa velikog broja računara, koji se nalaze pod kontrolom napadača, uz pomoć malicioznog softvera, u okviru mreže koja se naziva *botnet*, dok se sami računari u okviru *botnet*-a nazivaju *zombiji* ili *botovi*.

Napadači koji stoje iza *DDoS* napada često su motivisani različitim faktorima i razlozima. Ti razlozi su najčešće [6]:

- finansijska dobit: Ovaj razlog za napade predstavlja jedan od najčešćih, a istovremeno napadi pokrenuti finansijskom motivacijom su jedni od najopasnijih i najtežih da se zaustave. Ovi napadi najčešće su upereni protiv korporativnih mreža i zahtevaju veću količinu znanja i iskustva za pokretanje.
- ideološka uverenja: Ova motivacija za pokretanje napada takođe je vrlo česta, međutim ovi napadi najčešće nisu tehnološki napredni i lako se zaustavljaju.
- usporavanje performansi / prekid rada servisa: U ovom slučaju napadač hoće da napravi finansijsku štetu, time što će napasti kritični deo servisa (serveri banke,

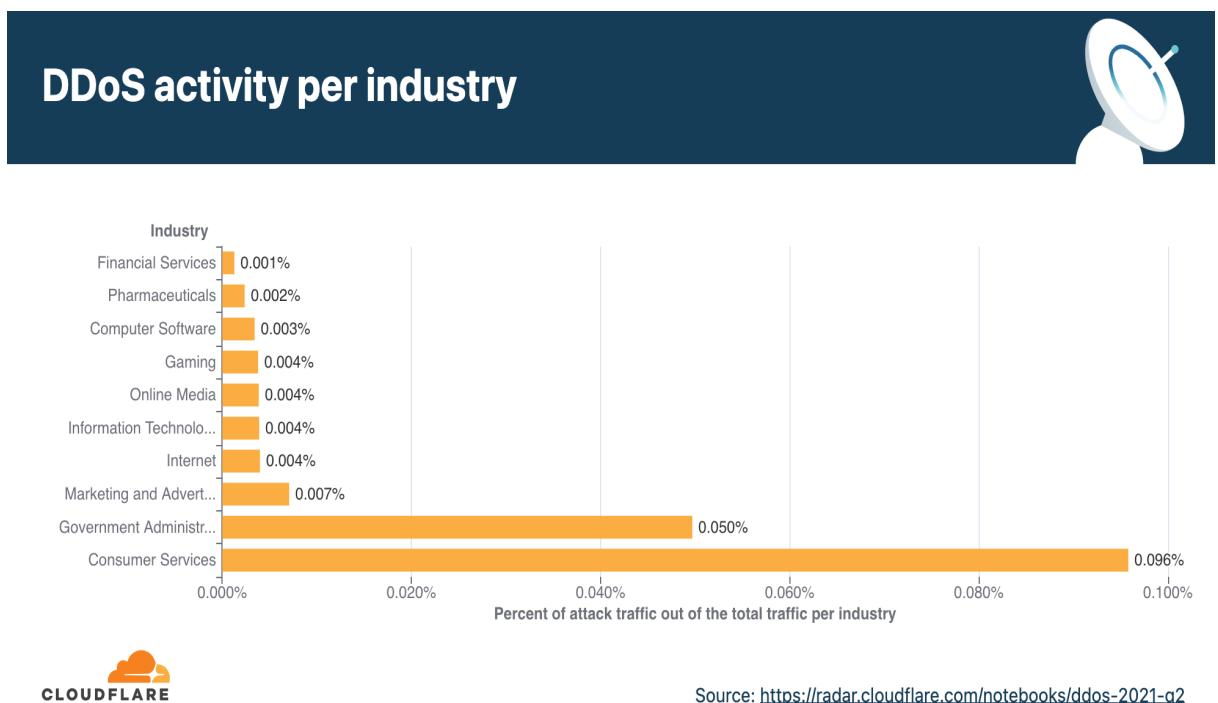
menjačnica virtuelnih valuta, servere koji se koriste za *online* video igre itd.).

- tzv. sajber ratovanje (eng. *cyberwarfare*): Napadači iz ove klase su uglavnom pripadnici vojske ili terorističkih organizacija i oni su politički motivisani da napadnu kritične sektore druge zemlje.

Dva bitna koncepta vezana za *DoS* i *DDoS* napade su *refleksija* i *amplifikacija*.

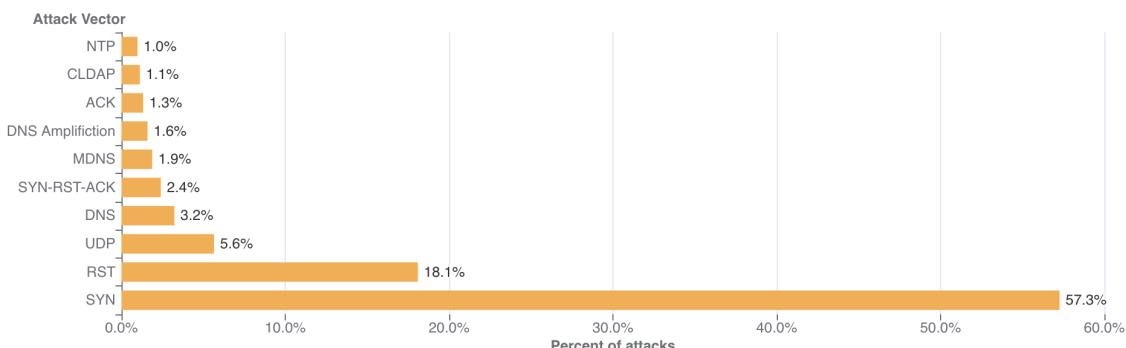
Refleksija je pojava gde se kod paketa koje napadač šalje radi *spoofing source IP* adresu paketa određenog protokola, tjst. ona se menja u adresu mete, tako da izgleda da je meta poslala te pakete i ovakvi paketi se prosleđuju refleksionim serverima, koji prime pakete, obrade ih na način specifičan za protokol i pošalju odgovor meti. [17]

Amplifikacija predstavlja pojavu povećavanja efektivnog mrežnog protoka koji napadač može da generiše napadom, najčešće tako što je odgovor mete (ili refleksionog servera, koji se preusmerava na metu), preko protokola koji se koristi u napadu, mnogo veći od paketa koji je napadač poslao. [17]



Slika 2.3 - Procenat DDoS saobraćaja zavisno od delatnosti kompanije u drugoj četvrtini 2021. god. [2]

Network-layer DDoS attacks: Distribution by top attack vectors



Source: <https://radar.cloudflare.com/notebooks/ddos-2021-q2>

Slika 2.4 – Najčešći tipovi DDoS napada u drugoj četvrtini 2021. god. [2]

Na slikama 2.3 i 2.4 prikazani su tipovi kompanija koje su najčešće bile cilj DDoS napada i najčešće metode napada koje su se koristile, u okviru istraživanja kompanije *Cloudflare* [2], koja između ostalog pruža usluge odbrane sajtova od DDoS napada, te raspolaže sa znatnom količinom podataka vezanim za napade.

2.4. KONCEPT BOTNET MREŽA

Botnet predstavlja mrežu uređaja na kojima se izvršava maliciozni kod, kojim se ostvaruje kontrola nad uređajem, koristeći tzv. *C&C* (*Command and Control*) kanal. Kroz *C&C* kanal centralni autoritet (tzv. *botmaster*) zadaje komande malicioznom softveru koji se izvršava na uređajima. Ove komande omogućavaju širok spektar operacija *botmaster*-u (naravno u zavisnosti od namene i implementiranih funkcionalnosti malicioznog programa), poput slanja *spam* poruka, krađu personalnih i finansijskih informacija, krađu kredencijala i kriptovaluta, pokretanje DDoS napada, propagaciju malicioznog koda na druge nebezbedne uređaje ili korišćenja hardverskih resursa uređaja radi ostvarivanja finansijske koristi kroz proces rudarenja kriptovaluta. [7]

Postoje tri glavna tipa *C&C* kanala [6][7]:

- kanali koji koriste *IRC* (*Internet Relay Chat*), protokol sedmog nivoa *OSI* modela, koji služi za komuniciranje putem slanja tekstualnih poruka, gde *botmaster* pošalje preko *IRC* protokola poruku koja dolazi do svih klijenata koji su konektovani na isti *IRC* server i nalaze se na istom *IRC* kanalu za slanje poruka, a botovi na isti način pošalju poruku odgovora. Naravno, ovde je potrebno naglasiti da *IRC* server ne mora da bude u kontroli *botmaster*-a, već on može da koristi i tuđu infrastrukturu za upravljanje nad *botnet*-om. Kredencijali za pristup odgovarajućem *IRC* serveru i kanalu su najčešće

hardkodovani u malicioznom programu.

- kanali koji koriste *HTTP* (*Hypertext Transfer Protocol*), protokol sedmog nivoa *OSI* modela, gde *botmaster* najčešće ima pristup *botnet* serveru u formi *web* sajta, preko koga uz pomoć odgovarajućih *HTTP* zahteva šalje komande koje ostanu zabeležene na serveru. Botovi koriste tehniku preuzimanja komandi gde povremeno pristupaju *botnet* sajtu sa odgovarajućim *HTTP* zahtevom i preko odgovora dobijaju odgovarajuće komande. U ovom slučaju *botmaster* mora da poseduje odgovarajuću serversku infrastrukturu. Odgovarajući domen ili *IP* adresa botnet servera su najčešće hardkodovani. Budući da je *HTTP* vrlo čest protokol ovo pruža dodatnu sigurnost malicioznom softveru i samom *botnet*-u.
- *P2P* (*peer-to-peer*) model komunikacije, koji funkcioniše najčešće korišćenjem *TCP* ili *UDP* protokola, gde ne postoji centralizovani server, a samim tim ne postoji kritična tačka *botnet* sistema (eng. *Single Point of Failure, SPOF*), a botovi (maliciozni softver koji se izvršava na njima) imaju komplikovaniju arhitekturu, koja omogućava svakom botu da ima funkcije *C&C* servera i da prosleđuje komande drugim botovima i da vraća odgovor. U ovom slučaju neophodno je da postoji odgovarajuća struktura negde na internetu, gde se održava dinamička lista botova sa kojima je moguće ostvariti početnu konekciju.

2.5. TIPOVI DDOS NAPADA

Postoji veliki broj različitih tipova *DDoS* napada. Glavnu razliku između pojedinih tipova napada predstavlja protokol koji se koristi za napad (najčešći su napadi koji koriste protokole četvrtog i sedmog nivoa *OSI* referentnog modela), da li je *DDoS* napad direktni ili se koriste refleksioni *host*-ovi, da li napad ima amplifikacioni efekat i da li napad koristi određenu slabost kod aplikativnog softvera mete.

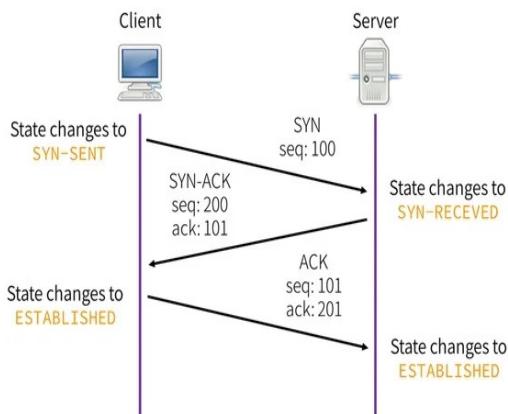
Naravno, potrebno je naglasiti da nijedan *DDoS* tip napada nije najbolji i najefektivniji u svakim slučajevima, već uglavnom napadači određuju tip napada koji bi bio najefektivniji u zavisnosti od svake pojedinačne mete, njenog tipa, protokola koje meta koristi, ali i same količine saobraćaja koju napadač može da izgeneriše, koja predstavlja vrlo bitan faktor.

Pregled najpoznatijih tipova *DDoS* napada dat je u nastavku:

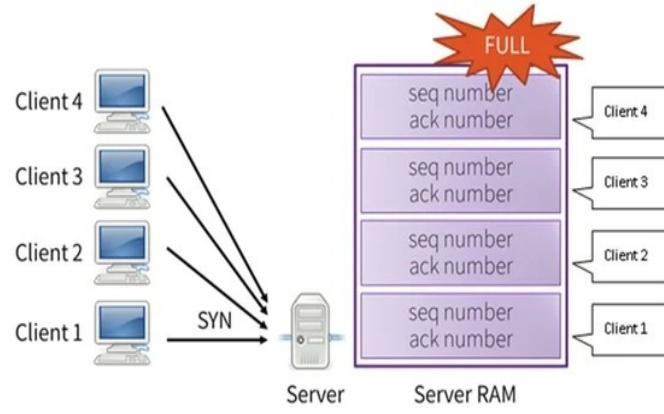
- *UDP flood*: Ovo je tip napada koji koristi *User Datagram Protocol* i njegovo svojstvo da je moguće slati podatke bez prethodno uspostavljene konekcije. Kod ovog napada šalju se datagrami uglavnom na *random* portove mete, što prouzrokuje zauzeće kapaciteta mrežnog linka za prenos i meta pokušava da detektuje da li postoji neka aplikacija koja radi i sluša pakete sa datog porta. Ponekad, ukoliko se ovakva

aplikacija ne detektuje, biće poslat *ICMP Echo Reply* paket *source IP* adresi koja je poslala *UDP* datagram, što dodatno doprinosi većem zauzeću mrežnog linka i usporavanju sistema mete. [10]

- *SYN flood*: Kod ovog tipa napada se koristi slabost *handshake* dela uspostavljanja konekcije *TCP* protokola. Koncept funkcionisanja *handshake* procesa je da prvo klijent inicira *TCP* konekciju slanjem *TCP* segmenta sa *set-ovanim SYN* flagom. Server na *SYN* segment odgovara sa *SYN-ACK* segmentom i kao odgovor očekuje *ACK* segment klijenta i nakon toga je *TCP* konekcija uspostavljena. Napadač u ovom slučaju šalje *SYN* segmente, server odgovara sa *SYN-ACK*, međutim napadač nikad ne pošalje *ACK* segment kojim se završava *handshake* deo i konekcija ostaje u poluotvorenom (eng. *half-opened*) stanju. Ovakva konekcija zauzima određeni deo resursa na meti i ostaje u poluotvorenom stanju dok ne prođe određeno vreme i ovo se dešava sve dok ima dostupnih resursa na meti. Kad ne bude više dostupnih resursa *TCP* konekcija neće moći da se uspostavi i tu se javlja *Denial of Service*. Na slikama 2.5 i 2.6 možemo videti ceo proces uspostavljanja *TCP* konekcije, kao i proces *SYN* napada i prevelikog iskorišćenja resursa mete. [11]



Slika 2.5 – Uspostavljanje *TCP* sesije [11]



Slika 2.6 – *SYN DDoS* napad [11]

- *ACK flood*: *ACK flood*, kao i *SYN flood*, zloupotrebljava isti protokol – *TCP*. On se koristi isključivo protiv uređaja koji moraju da obrade svaki paket koji prime – *firewall* sistemi i serveri. Napadač će poslati veliku količinu podataka fragmentovanih na pakete i na taj način se zauzima velika količina mrežnog protoka i sistemskih resursa mete. [12]
- *RST flood*, *FIN flood* i *SYN ACK flood*: Ova tri napada su vrlo slična *SYN* napadu, samo što napadač šalje segmente sa *set-ovanim RST*, *FIN* ili *SYN* i *ACK* flagovima. U okviru sva 3 napada cilj je da se obradom ovih segmenata zauzme procesorsko vreme i

drugi resursi mete. [12]

- *HTTP flood*: *HTTP*, protokol za prenos sadržaja *web* stranica, predstavlja jedan od najkorišćenijih protokola na internetu. Ovaj protokol se najčešće koristi za napad na *web* servere koji služe za *host*-ovanje sajtova na internetu. U okviru napada šalje veliki broj poruka sa *HTTP GET* metodom, koja se koristi za preuzimanje određenih resursa (teksta, slika, fajlova itd.) sa servera ili *HTTP POST* metodom, kojom se šalju podaci na *HTTP* server, najčešće radi kreiranja ili dodavanja dodatnih resursa, što zahteva procesorsko vreme na serveru da se obradi. *HTTP GET flood* se može smatrati *DDoS* napadom sa amplifikacijom, budući da je veličina *HTTP GET request*-a generalno mnogo manja od *HTTP GET response* poruke koja se dobija od servera. [12]
- *XML-RPC napad*: *WordPress* predstavlja jednu od popularnih platformi sa otvorenim kodom (eng. *open-source*) za kreiranje *web* sajtova. Implementiran je u *PHP* jeziku i koristi *MySQL* kao bazu podataka, a izmena sadržaja *web* sajta je vrlo jednostavna preko *CMS*-a (eng. *Content Management System*) [14]. *XML-RPC* predstavlja sistem pozivanja udaljenih metoda (eng. *Remote Procedure Call*), koji koristi *HTTP* kao protokol za transport poruka i *XML* kao mehanizam za enkodiranje podataka, koji se koristi upravo kod *WordPress* sajtova u obliku fajla *xmlrpc.php*. Ovaj zastareo sistem omogućava da administratori mogu da upravljaju svojim sajtom preko uređaja koji nisu računari (mobilni telefoni, tableti itd.). Danas se može isključiti koristeći različite dodatke na *WordPress*, a kao alternativa u budućnosti je predviđeno korišćenje novog *API* sistema umesto *XML-RPC*. [15] Jedna od funkcija *XML-RPC* sistema je i *pingback*, koji omogućavaju *link*-ovanje objave na jednom sajtu sa *onem* na drugom, kao što se može detaljnije videti na [16], gde je najbitniji detalj to da se referencirani sajt otvara od strane sajta na kome se izvršio *pingback*. Napadači tipično skeniraju internet tražeći sajtove koji funkcionišu preko *WordPress* platforme i zatim pokreću *bruteforce* napad na dobijenu listu sajtova. Kao rezultat *bruteforce* napada dobija se lista sajtova gde napadači imaju pristup *XML-RPC* komponenti. Koristeći ovaj pristup, napadači preko *pingback* funkcije započinju *DDoS* napad koji ima karakteristike amplifikacije i refleksije. Preko ove metode čak i bez *botnet*-a je moguće napraviti velike probleme u funkcionisanju sajtova.
- *ICMP flood*: Kod ovog napada se koristi *ICMP* (*Internet Control Message Protocol*), protokol trećeg nivoa *OSI* referentnog modela, koji se generalno koristi za dijagnostiku, ispitivanje mreže i za slanje greški vezanih za *IP* operacije. Dakle, napadač šalje veliki broj *ICMP echo request* paketa, a kao odgovor od mete dobija *ICMP echo reply* pakete, te na taj način se troši mrežni kapacitet linka koji meta

koristi u oba smera. Najlakši način odbrane od ovakvih napada je zabraniti korišćenje *ICMP* protokola na uređajima koji su napadnuti. [12]

- *DNS napad:* *DNS (Domain Name System)*, protokol sedmog nivoa *OSI* modela, predstavlja jedan od najbitnijih protokola interneta, budući da on omogućava razrešavanje domena u *IP* adresu. *DNS* kao protokol prenosa na transportnom sloju koristi *UDP*, zato što su tipično upiti i odgovori male veličine. Vremenom postali su sve češći napadi koji koriste ovaj protokol, bilo u obliku *DNS flood* napada gde se velikim brojem *DNS query* upita vrši napad na *DNS* servere ili u obliku gde se *DNS* koristi za napade koji imaju refleksioni i amplifikacioni karakter, koji predstavlja češći oblik i kojim je moguće napasti metu koja nije *DNS* server. U okviru ovog napada, zloupotrebljava se uloga tzv. *DNS Open Resolver-a*, koji predstavljaju *DNS* servere gde je moguće poslati *query* i dobiti odgovor za bilo koji *DNS* domen, bez provere od koga je *query* dobijen. Napad se realizuje masovnim slanjem *DNS* upita ka velikoj listi *DNS Open Resolver-a*, gde se radi tzv. *spoofing source* adrese u okviru *DNS* upita, tako da se tu upiše adresa mete. Na taj način meta dobija odgovore na *DNS* upite koje nije poslala, koji crpe njene resurse i dovode do prestanka rada. [17]
- *NTP napad:* *NTP (Network Time Protocol)* predstavlja protokol aplikacionog nivoa *OSI* modela, koji služi za sinhronizaciju vremena između računarskih sistema i predstavlja jedan od najstarijih protokola koji se koriste na internetu. Kao transportni protokol koristi *UDP*. *NTP* tip napada predstavlja refleksioni i amplifikacioni *DDoS* napad i on je postao jedan od najpopularnijih napada oko 2013. godine. Za refleksiju *NTP DDoS* koristi *NTP* servere koji dozvoljavaju korišćenje *MONLIST* komande, kojom se vraća poslednjih 600 klijentskih adresa koje su ostvarile konekciju sa tim *NTP* serverom. *NTP request* poruka za *MONLIST* komandu koji napadač šalje ima veličinu oko 64 B, dok odgovor koji se dobija može da bude veličine od oko 48 200 B, koji su raspoređeni na više odgovora, što govori o značajnom nivou amplifikacije kod ovog napada. Napad se preusmerava na metu tako što se radi *spoofing source IP* adrese *NTP request* poruke, tako da *NTP* server pošalje odgovor meti napada. Na osnovu istraživanja kompanije *Arbor Networks*, 2014. godine 85% *DDoS* napada veličine preko 100 Gbps su koristili upravo *NTP*, budući da je u aprilu 2014. broj *NTP* servera pogodnih za slanje napada bio oko 430,000. Ovaj broj je kasnije krenuo brzo da pada, a sa njime i frekvencija *NTP DDoS* napada. [18]
- *CHARGEN napad:* *CHARGEN (Character Generation)* protokol predstavlja jedan od najstarijih protokola. On služi za debagovanje, testiranje i za potrebe merenja, međutim danas se retko koristi. Definisan je u RFC 864 [19] i kao protokol na

četvrtom sloju *OSI* modela može da koristi i *TCP* i *UDP*. Kada klijent pošalje serveru *CHARGEN* zahtev server odgovara sa *reply* porukom koja je dužine od 0 do 512 B. Zahtev je obično vrlo male veličine, pa je preko ovog protokola moguće postići veliki stepen amplifikacije ukupne količine saobraćaja *DDoS* napada. Pored amplifikacionih efekata, *DDoS* koji koristi ovaj protokol je naravno i refleksione prirode, budući da se *source IP* adresa *CHARGEN request* poruke menja u adresu mete. [20]

- *Ping of death:* *Ping of death* predstavlja zastareo koncept još jednog napada zasnovanog na *ICMP*. U okviru ovog napada, šalje se *ICMP echo reply* paket koji je veličine preko maksimalno dozvoljene za paket (65,535 B, definisano u RFC 791 [21]). Naravno ovakav paket pre slanja će biti fragmentovan na više delova. Ukoliko meta nema zaštitu protiv ovakve vrste napada, događa se *buffer overflow* prilikom procesa sklapanja paketa i to može dovesti do pada sistema ili čak do injekcije malicioznog koda u sistem mete. [22]
- *Smurf napad:* Ovaj zastareli tip napada takođe koristi *ICMP*, tako što pošalje *ICMP echo request* paket gde je *source* adresa *IP* adresa mete, a *destination* adresa je *broadcast* adresa rutera u mreži. Kad ruter primi ovakav paket, proslediće ga do svakog *host-a* u mreži i svaki *host* će da pošalje *ICMP echo reply* paket meti i na taj način će mu preopteretiti mrežni link i iskoristiti resurse na meti za obradu tih paketa. [23]

3. IMPLEMENTACIJA APLIKACIJE

U okviru ovog poglavlja dati su implementacioni detalji vezani za konstrukciju aplikacije rešenja – korišćeno implementaciono i razvojno okruženje, programski jezik i detalji vezani za implementaciju dizajna i različitih funkcionalnosti aplikacije.

3.1. RAZVOJNO I IMPLEMENTACIONO OKRUŽENJE

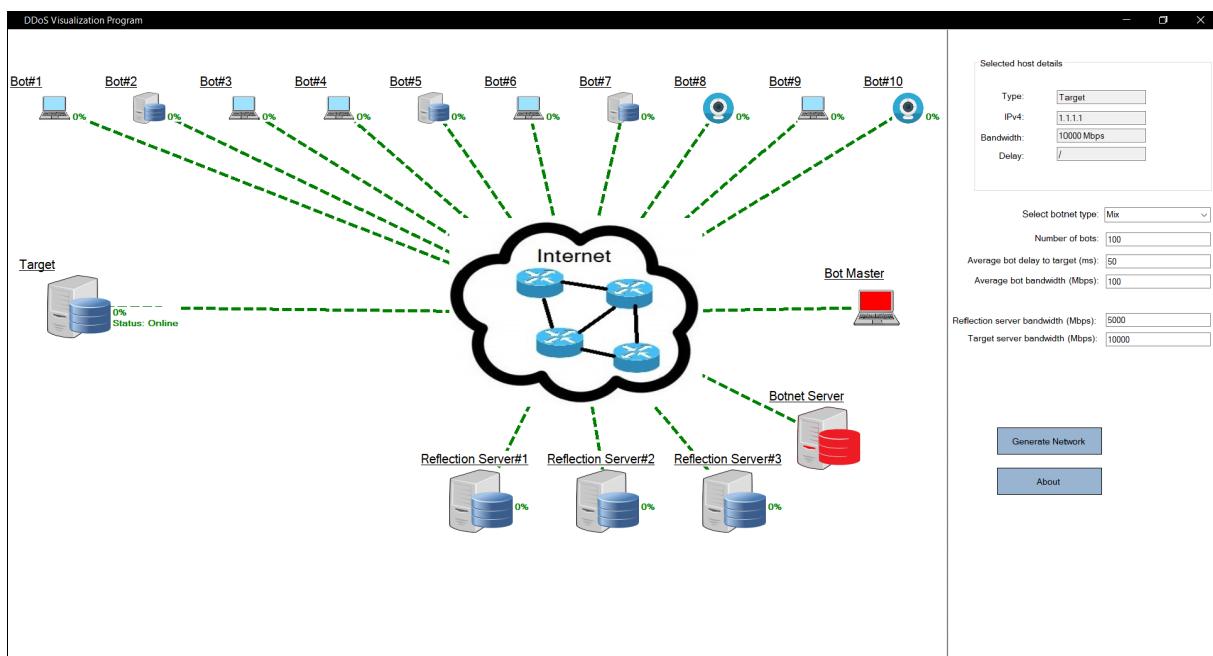
Kao razvojno okruženje korišćen je *Microsoft Visual Studio 2015* [32], dok je za implementaciono okruženje korišćen *.NET Framework verzija 4.5* [33] [34], koje predstavlja softverski *framework* koji je razvio *Microsoft* i namenjen je izvršavanju primarno na *Windows* platformi, sa podrškom za izvršavanje na ostalim platformama. Programski kod se izvršava u *CLR (Common Language Runtime)* okruženju, koje je slično virtuelnoj mašini za *Javu (Java Virtual Machine, JVM)*. Ovo okruženje brine o dealokaciji memorije preko *garbage* kolektora, pruža podršku za izuzetke u izvršavanju programa, odličan interfejs za razvoj *GUI (Graphical User Interface)* aplikacija, podršku za kriptografske algoritme, mrežnu komunikaciju, povezivanje sa bazom podataka itd.

Aplikacija je razvijena koristeći *Visual Basic .NET* (skraćeno *VB.NET*) [24], koji predstavlja objektno-orientisani programski jezik koji zajedno sa *C#* i *F#* predstavlja jedan od tri glavna jezika za *.NET* platformu. *VB.NET* je nastao kao naslednik *VB6* jeziku 2002. godine. Konceptualno, jezik je vrlo sličan *Javi*, dok sintaksa ima sličnosti sa *Python* jezikom.

3.2. DIZAJN APLIKACIJE

Aplikacija je dizajnirana tako da se sastoji od više formi. Glavna forma, ujedno i početna forma aplikacije se sastoji iz dva panela, koji su podeljeni preko komponente koja se naziva *SplitContainer*, koja omogućava efikasnu podelu ekrana na dva dela, a veličina svakog panela može da se pomera, selektovanjem i pomeranjem linije koja razdvaja panele. Ideja iza ovakvog pristupa je to da aplikacija može da se menja od strane korisnika u zavisnosti kako mu odgovara, iako je podrazumevano *fullscreen* aplikacija može da radi i u režimu prozora upravo koristeći ovu komponentu. Scroll barovi se automatski generišu ukoliko za njima ima potrebe, tjst. ukoliko sadržaj ne može da stane u odgovarajući panel. U levom panelu se nalazi radna površina, na kojoj će biti prikazana cela mreža koja je generisana. Mreža je predstavljena vizuelnim prikazom *host-ova* (slika odgovorajućeg tipa *host-a*, labele za identifikaciju i za prikaz procentualnog iskorišćenja mrežnog protoka), prikazom interneta i

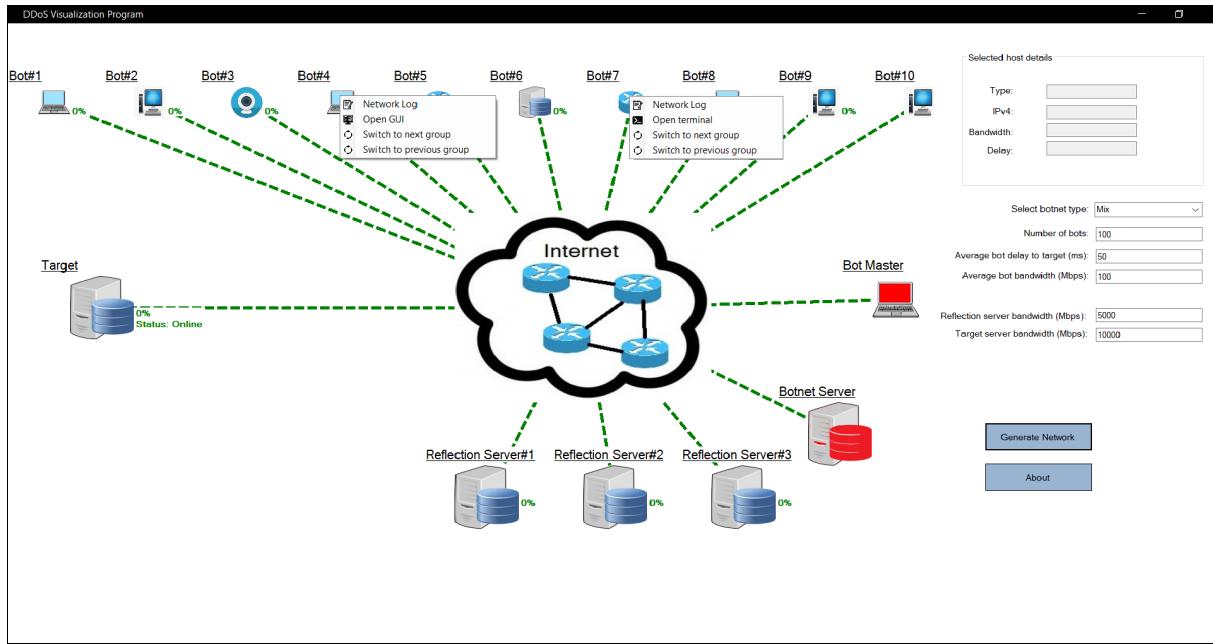
linkovima koji povezuju *host*-ove sa internetom, koji omogućava komunikaciju među *host*-ovima. Budući da se linkovi crtaju uz korišćenje grafike forme, neophodno je da se svaki put kad se dogodi *Paint* event linkovi ponovo iscrtaju. Desni panel u okviru forme predstavlja konfiguracioni panel, u kome se nalaze *TextBox* i *Label* komponente, uz pomoć kojih se zadaje konfiguracija programu, kao parametri za generisanje mreže i *ComboBox* preko koga se bira tip *botnet*-a, odnosno uređaja koji se nalaze u *botnet*-u. Takođe, u okviru desnog panela nalazi se i *GroupBox* komponenta u okviru koje se prikazuju detalji za selektovanog *host*-a. Prikaz glavne forme, u trenutku kad je generisana mreža dat je na slici 3.1.



Slika 3.1 – Glavna forma aplikacije sa generisanim mrežom

U okviru radne površine, *host*-ovima može da se menja pozicija, tako što se selektuje slika želenog *host*-a i prevuče levim klikom na novu poziciju.

Desnim klikom na *host*-a se otvara meni, koji je realizovan korišćenjem komponente *ContextMenuStrip*, koja se dinamički generiše svaki put kad se detektuje desni klik i u isti meni se stavljuju odgovarajuće opcije u zavisnosti od tipa *host*-a koji je selektovan. U okviru ovog menija svaka opcija ima svoju ikonicu, koja je vizuelno povezana sa njenom ulogom. Svaki bot, kao što je prikazano na slici 3.2, ima opciju da otvorи svoj mrežni *log*, svoj interfejs (grafički ili terminal), a takođe i postoje opcije koje služe za promenu grupe botova koja se prikazuje. *Switch to next group* radi promenu prikaza na grupu od 10 sledećih botova, dok *switch to previous group* prikazuje prethodnih 10 botova. Naravno, ove operacije funkcionišu kružno, tako da ukoliko sa prvih 10 botova mi pritisnemo operaciju za prethodnu grupu, kao rezultat pojavilo bi se poslednjih 10 botova na ekranu. Ista stvar ukoliko imamo poslednjih 10 botova na ekranu i pritisnemo opciju za sledeću grupu – dobili bi prvih 10 botova.



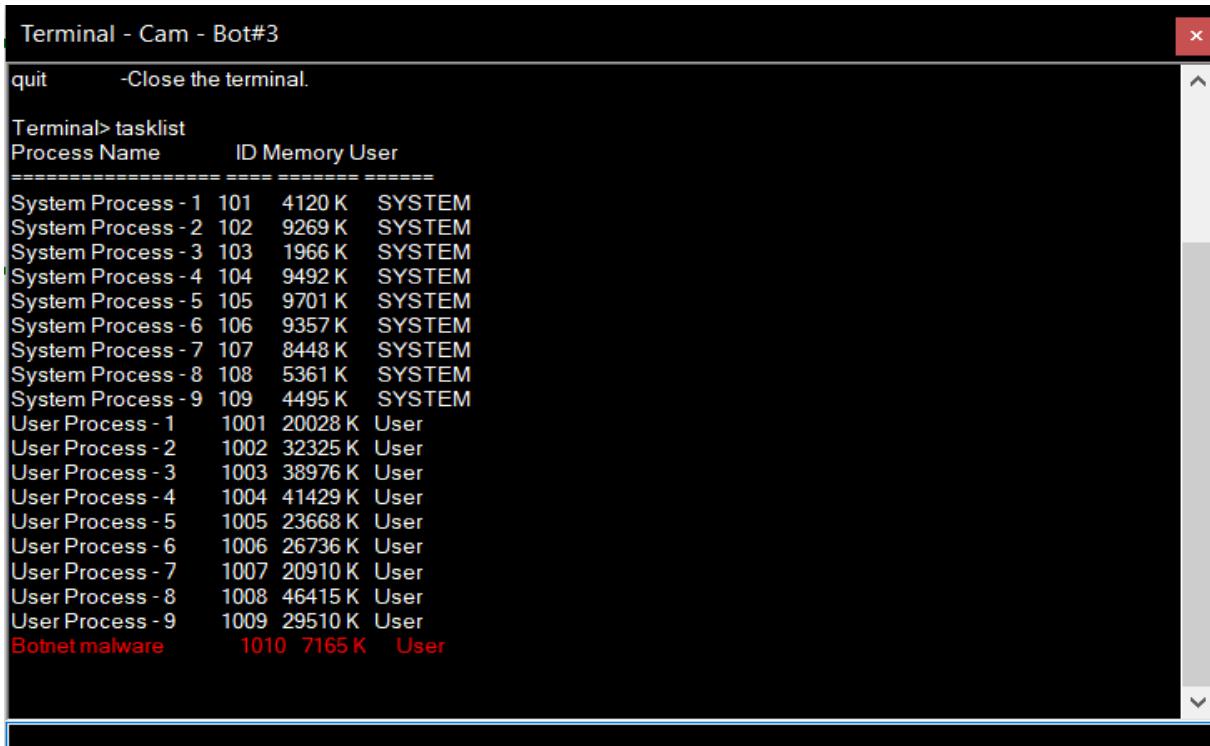
Slika 3.2 – Glavna forma aplikacije, sa prikazanim opcijama za hostove

Na slici 3.3 prikazana je forma koja predstavlja GUI okruženje, kad je otvoren prozor *Process Manager* u okviru forme. U okviru ove forme, razdvajanje na prozore se vrši korišćenjem *TabControl* komponente. Prozor *Process Manager* je realizovan korišćenjem *ListView* komponente, uz pomoć koje je moguće kreirati tabelu sa više atributa, za prikaz nizova podataka.

Process Name	Process ID	Memory	User
System Process - 1	101	3882 K	SYSTEM
System Process - 2	102	9696 K	SYSTEM
System Process - 3	103	2442 K	SYSTEM
User Process - 1	1001	48519 K	User
User Process - 2	1002	48557 K	User
User Process - 3	1003	25242 K	User
User Process - 4	1004	18392 K	User
User Process - 5	1005	33262 K	User
User Process - 6	1006	10965 K	User
User Process - 7	1007	13016 K	User
User Process - 8	1008	43055 K	User
User Process - 9	1009	14549 K	User
User Process - 10	1010	15228 K	User
User Process - 11	1011	26133 K	User
Botnet malware	1012	6387 K	User

Slika 3.3 – GUI okruženje, prikaz liste procesa kod odabranog uređaja

Na slici 3.4 je prikazano komandno okruženje terminala, kad se izvrši komanda za prikaz trenutnih procesa na posmatranoj web kameri. Ono je realizovano koristeći *RichTextBox* komponentu, zbog podrške za korišćenjem više boja u okviru teksta.

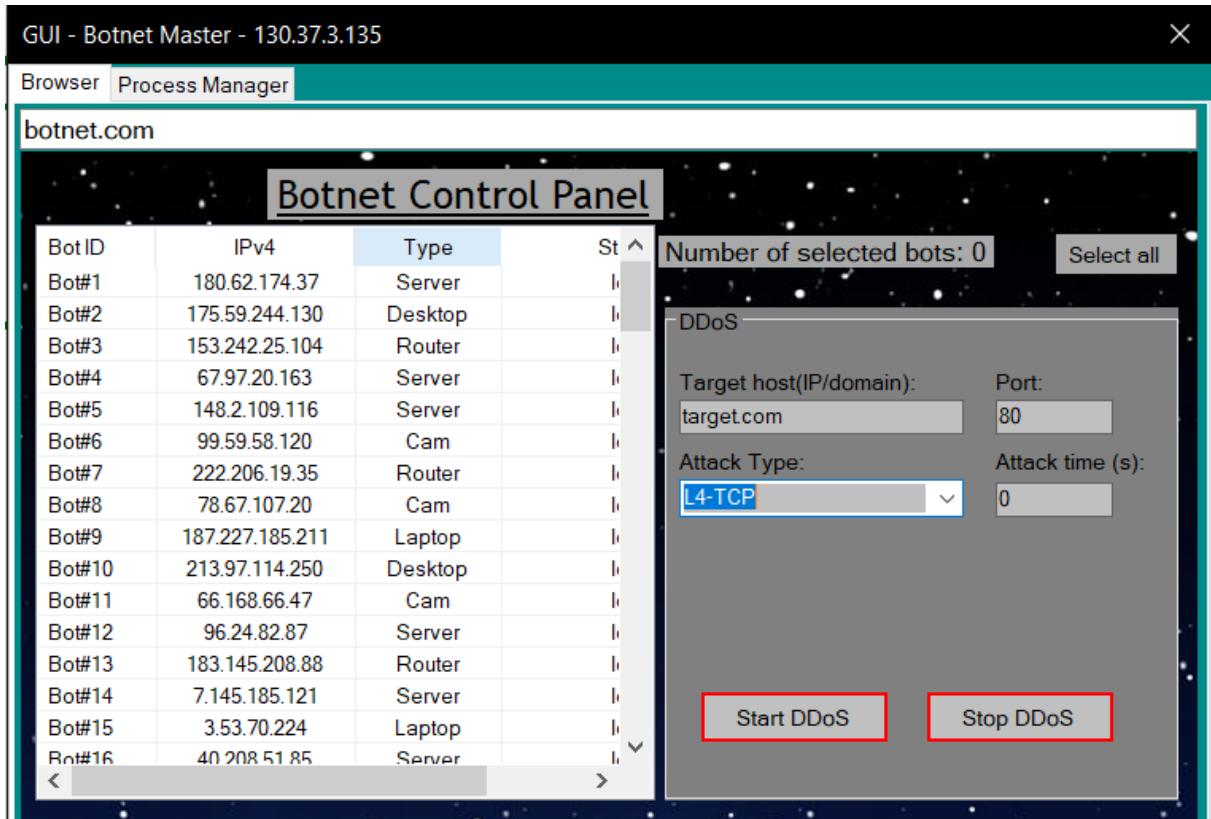


```
Terminal - Cam - Bot#3
quit      -Close the terminal.

Terminal> tasklist
Process Name      ID Memory User
=====
System Process - 1 101   4120 K  SYSTEM
System Process - 2 102   9269 K  SYSTEM
System Process - 3 103   1966 K  SYSTEM
System Process - 4 104   9492 K  SYSTEM
System Process - 5 105   9701 K  SYSTEM
System Process - 6 106   9357 K  SYSTEM
System Process - 7 107   8448 K  SYSTEM
System Process - 8 108   5361 K  SYSTEM
System Process - 9 109   4495 K  SYSTEM
User Process - 1 1001  20028 K User
User Process - 2 1002  32325 K User
User Process - 3 1003  38976 K User
User Process - 4 1004  41429 K User
User Process - 5 1005  23668 K User
User Process - 6 1006  26736 K User
User Process - 7 1007  20910 K User
User Process - 8 1008  46415 K User
User Process - 9 1009  29510 K User
Botnet malware     1010  7165 K  User
```

Slika 3.4 – Terminal, prikaz rezultata komande tasklist

Na slici 3.5 dat je grafički prikaz HTTP kontrolnog panela za *botnet*, preko koga se zadaju komande za pokretanje i zaustavljanje *botnet* napada. Tip *DDoS* napada se bira koristeći komponentu *ComboBox*, dok je lista botova koji su *online* prikazana u okviru *ListView* komponente, zajedno sa ostalim detaljima. Korisnik započinje napad tako što bira botove koje želi, koristeći dugme *Select all* ili tako što uradi višestruko biranje, držeći taster *CTRL* na tastaturi i koristeći desni klik. Napad se započinje i prekida koristeći odgovarajuće tastere. Broj selektovanih botova se može videti koristeći labelu *LBLSelected*, koja se menja svaki put kad se detektuje događaj promene broja selektovanih botova u okviru *ListView* komponente. Svaki put kad se pokrene ili zaustavi *DDoS* napad vrši se deselekcija svih selektovanih botova. Ukoliko se unese pogrešna vrednost nekog od parametara ili se koriste parametri mete koje program ne podržava (bilo šta osim servera mete – target.com ili IP adrese 1.1.1.1), program će prikazati odgovarajuću poruku greške korisniku.



Slika 3.5 – Prikaz kontrolnog panela botnet-a

3.3. IMPLEMENTACIJA FORMI U OKVIRU APLIKACIJE

U nastavku su opisane glavne forme u okviru aplikacije i njihova implementacija:

- Glavni prozor aplikacije (klasa *MainWindow*): Ova forma predstavlja početnu i ujedno i glavnu formu aplikacije. U okviru nje, pored vizuelnih elemenata i tekstualnih polja za konfiguraciju, nalaze se implementirane metode za vizuelizaciju i generisanje konfiguracije cele mreže koja se prikazuje. Metoda *Draw_Network* se poziva kad se dogodi događaj iscrtavanja forme i ona je nadležna za pozivanje odgovarajućih metoda za iscrtavanje mrežnih konekcija između slika koje predstavljaju *host*-ove u mreži. *BTNGenerate_Click* je metoda koja se poziva kada se registruje klik na *Generate Network* dugme i ova metoda proverava date konfiguracione parametre i kreira prikaz računarske mreže u skladu sa njima. *BTNAbout_Click* predstavlja metodu koja se poziva pri kliku na *About* dugme i ovom metodom se prikazuje *Splashscreen* sa informacijama o mentoru i autoru aplikacije. *Generate* metoda je implementirana tako da postavlja uvek iste koordinate *host*-ovima i internetu, radi preglednosti. Mrežni protok svakog bota i njegov *delay* do servera mete se generišu kao nasumični celi brojevi od 1 do granice koju je korisnik postavio. Ova raspodela bi trebala da bude uniformna, međutim zavisi od implementacije generatora pseudoslučajnih brojeva u okviru

Random klase implementacionog okruženja.

- Grafički interfejs (klasa *GUI*): *GUI* predstavlja grafički korisnički interfejs koji se koristi kod servera, laptopova i računara. Kod ove forme su implementirane metode: *SetHost* i *SetBotMaster*, koje regulišu za koji uređaj je data forma otvorena, *EnterWebsite*, koja se poziva kad se pritisne *Enter* taster, kad se unese *IP* adresa ili *URL* sajta i koja služi da prikaže sajt koji se otvara ili grešku ukoliko ne može da se otvori i *makeProcessList*, metoda koja generiše nasumično listu procesa na datom *host*-u koji se prikazuju u okviru *Process Manager* prozora.
- Komandna linija (klasa *Terminal*): U okviru ove klase, koja predstavlja implementaciju *Linux* terminala za *host*-ove koji su ruteri ili *web* kamere, implementirane su sledeće metode: *SetHost*, metoda gde se reguliše za koji uređaj je data forma pokrenuta, *EnterCommand*, koja se poziva kada se unese komanda i koja dalje poziva *HandleCommand*, koja predstavlja glavnu metodu u okviru ove klase, gde se obrađuju različite komande, poput pinga ili tasklist komande.
- Mrežni log (klasa *NetworkLog*): U ovoj klasi su implementirane metode *SetHost* i *SetServer*, koje postavljaju za koji uređaj je forma otvorena. Takođe implementirane su metode koje omogućavaju refreshovanje loga, brisanje loga i zatvaranje prozora.

3.4. IMPLEMENTACIJA SAJTOVA

Web sajtori u okviru aplikacije su implementirani koristeći napravljenu klasu *Site* kao šablon. U okviru ove klase, koja u osnovi predstavlja formu bez definisanih granica (polje *BorderStyle* u Visual Studio okruženju) je definisan panel nazvan *Website* i koji je po pristupu javan (polje *Modifiers* u Visual Studio okruženju). U okviru ovog panela koji je veličine 740 x 420 se definiše izgled sajta, njegove komponente itd. U okviru klase *Site* postoji takođe i prazna metoda *WebsiteLoad* koja se nadjača u klasi sajta koji se pravi i poziva se prilikom učitavanja sajta. Kad se učita sajt, u formu *Browser*-a se učita dati panel, a svi *event*-ovi koji su definisani u okviru klase sajta i dalje funkcionišu.

Ovakva implementacija je izabrana da bi se zadržalo korišćenje dizajnera formi u okviru razvojnog okruženja, dok istovremeno se omogućava polimorfizam u smislu kreiranja sajtova i dodeljivanja sajtova *web* serverima koji ih hostuju. Da bi se dodao sajt u okviru Visual Studio okruženja, napravi se nova *Windows* forma, potom se pritisne opcija *Show All Files* u *Solution Explorer* prozoru i zatim se u okviru *.Designer.vb* fajla novonapravljene klase sajta izmeni linija `Inherits System.Windows.Forms.Form` u `Inherits Site`, kao što je prikazano na slici 3.6.

```

TestWebsite.Designer.vb + x
DDoS Visualization Program - TestWebsite - Dispose
1 <Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _ 
2     1 reference
3     Partial Class TestWebsite
4         Inherits Site
5             'Form overrides dispose to clean up the component list.
6             <System.Diagnostics.DebuggerNonUserCode()> _
7                 Protected Overrides Sub Dispose(ByVal disposing As Boolean)
8                     Try
9                         If disposing AndAlso components IsNot Nothing Then
10                             components.Dispose()
11                     End If
12                     Finally
13                         MyBase.Dispose(disposing)
14                     End Try
15                 End Sub
16
17             'Required by the Windows Form Designer
18             Private components As System.ComponentModel.IContainer
19
20             'NOTE: The following procedure is required by the Windows Form Designer.
21             'It can be modified using the Windows Form Designer.
22             'Do not modify it using the code editor.
23             <System.Diagnostics.DebuggerStepThrough()> _
24             Private Sub InitializeComponent()
25                 components = New System.ComponentModel.Container
26                 Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
27                 Me.Text = "TestWebsite"
28             End Sub
29         End Class

```

Slika 3.6 – Prikaz procesa kreiranja klase novog sajta u okruženju Visual Studio 2015

Sajt se serveru zadaje prilikom instanciranja, u okviru konstruktora i ne može se menjati. Sajtovi koji su već implementirani su:

- *ErrorSite*, kojim se interno služi *Browser* iz *GUI* okruženja *host*-ova, da prikaže grešku, ukoliko sajt nije pronađen ili ga nije moguće učitati zbog *DDoS* napada.
- *TargetSite* (domen target.com ili IP adresa 1.1.1.1), koji predstavlja primer sajta koji koristi meta napada.
- *Botnet* (domen botnet.com ili IP adresa 2.2.2.2), koji predstavlja sajt koji se nalazi na serveru koji predstavlja server *botnet* mreže i na kome se nalazi kontrolni panel za pokretanje i zaustavljanje napada, kao i za pregled trenutnih aktivnosti botova. Za pristupanje kontrolnom panelu, potrebno je ulogovati se prethodno. Ovo je implementirano koristeći panel koji prekriva sajt, dok se admin *botnet* mreže ne uloguje.

Sajtovi se otvaraju preko deljene globalne *asinhrona* funkcije *openWebsite* u okviru klase *Server*, gde se vrši pretraga sajta na osnovu date *URL* adrese i vraća instanca odgovarajućeg sajta u odgovarajući *Browser* koji je pozvao metodu otvaranja sajta, ukoliko sajt postoji i njegovo otvaranje je moguće, a u suprotnom se vraća *ErrorSite* sa odgovarajućom porukom. Funkcija je implementirana kao asinhrona, budući da ne mora da odmah vraća rezultat već ima ugrađena vremena čekanja da se sajt pronađe ili da se proba konekcija ka njemu.

3.5. IMPLEMENTACIJA DDOS NAPADA

Botnet preko koga se kontrolišu botovi je implementiran kao *botnet HTTP* tipa. Mogućnost pokretanja *DDoS* napada se vrši sa sajta *botnet*-a (*botnet.com* ili IP adresa 2.2.2.2), kome se pristupa preko *GUI* forme i *Browser* prozora. Korisnički kredencijali za logovanje u kontrolni prozor *botnet*-a su **admin** za korisničko ime i **123** za lozinku. *DDoS* napad se pokreće biranjem određenog tipa *DDoS* napada, *IP* ili domena mete, porta koji se napada na meti, specifikacijom trajanja napada u sekundama i biranjem određenog broja botova koji će da izvrše napad. Ukoliko se zadaju loši parametri ili se navede *host* koji nije server mete prikazuje se poruka greške.

Ukoliko je vreme napada 0, botovi će napadati dok im se ne zada komanda za zaustavljanje napada. Svaki bot može da napada samo jednu metu istovremeno.

Metoda *BTNStart_Click*, kojom se pokreće napad pritiskom na dugme *Start DDoS*, u okviru *botnet*-a je implementirana kao *asinhrona* metoda, tako da ne dolazi do blokiranja cele forme dok data metoda čeka da prođe trajanje napada, da bi isključila botove koji su započeli napad preko ove metode. Da bi se zapamtili botovi koristi se lokalni red botova, a da ne bi došlo do problema, iako teorijski je vrlo teško da se dogodi, koristi se deljeni *Mutex* objekat u okviru forme nad kojim se radi *SyncLock* pre izvršavanja operacija koje modifikuju mrežne kapacitete i ostale parametre *host*-ova, u zavisnosti od napada.

Napad se može zaustaviti pritiskom na dugme *Stop DDoS*, što poziva metodu *BTNStop_Click*, kojom se prekida napad za selektovane botove. Treba imati u vidu da je metoda za pokretanje napada implementirana tako da ukoliko dođe do manuelnog zaustavljanja napada, pa se ponovo pokrene napad sa istim botovima, može da dođe do toga da se napad na botovima zaustavi prevremeno, zbog prethodnog vremenskog limita.

U nastavku je dat pregled implementiranih *DDoS* napada u okviru aplikacije i detalji vezani za njihovo funkcionisanje:

- 1) *TCP flood* – Kod ovog napada botovi koji su selektovani za napad u okviru *botnet*-a pokreću zahteve za *TCP* konekcijom na port koji je zadat od strane *botmaster*-a i kad se konektuju kreću da šalju pakete koji predstavljaju besmislen saobraćaj, čiji je cilj zaguši mrežni link servera mete. Ovaj napad funkcioniše samo ako postoji aplikacija koja sluša na datom *TCP* portu, zbog prirode *TCP* protokola. Ovaj napad, iako danas nije čest, budući da se lako detektuje i blokira, implementiran je koristeći *LOIC* (*Low Orbit Ion Cannon*) alatku i njen *TCP* napad kao referencu. Bot koristi ceo kapacitet linka koji ima na raspolaganju i taj kapacitet se dodaje ukupnom saobraćaju koji pristiže do servera. Ukoliko ukupan saobraćaj koji dolazi do servera pređe kapacitet njegovog mrežnog linka, tada meta odlazi *offline* i nije moguće pristupiti sajtu mete.

- 2) *UDP flood* – Ovaj napad je vrlo sličan sa pogleda implementacije kao *TCP*, međutim ne zahteva da dati port bude otvoren i da na njemu postoji aplikacija koja sluša, budući da je *UDP* beskonekcioni protokol. Bot koristi kapacitet celog mrežnog linka koji ima na raspolaganju za napad.
- 3) *SYN flood* – Ovaj napad, zbog njegove specifičnosti, implementiran je koristeći niti. Naime, kod servera mete, koristi se konstanta *PortLimit* koja iznosi 64000 i koja označava maksimalan broj poluotvorenih konekcija na meti, pre nego što joj ponestane resursa i meta ode *offline*. Za maksimalno vreme čekanja da stigne poslednji *ACK* (koji zbog prirode *SYN* napada nikada neće biti poslat) je uzeta konstanta od 5 sekundi. Postoji nit kod servera mete koja obrađuje čekanje na *ACK* segment, tako što svake sekunde doda određeni broj novih segmenata u red, na osnovu brzine *SYN* napada. Ukoliko taj broj premaši određenu granicu, server mete postaje nedostupan. Kao brzina slanja napada svakog *host*-a uzeta je konstanta 200 zahteva po sekundi, dok je mrežni protok potreban za izvođenje ovakvog napada 1 *Mbps* za svakog *host*-a.
- 4) *HTTP GETflood* – U okviru implementacije ovog napada, zahtevano je da sajt koji se napada bude *web* server i da kao port se bira port 80 koji koristi *HTTP*, u suprotnom napad neće imati efekta. Ukoliko se pokrene ovakav napad na server mete, napad se ponaša vizuelno vrlo slično kao *TCP* napad. Međutim zbog amplifikacione prirode napada, u ovom slučaju botovi će da koriste samo jedan procenat svog linka za napad u smeru ka meti, gde šalju *HTTP GET* poruke, dok će ostatak biti iskorišćen za odgovor servera ka botu, u obliku *HTTP* sadržaja *web* stranice. Budući da je prikazana iskorišćenost linka u procentima za oba smera prenosa podataka, ovaj detalj neće moći da se vidi u toku napada.
- 5) *XML-RPC napad* – U okviru ovog napada, botovi selektovani za napad šalju *pingback* poruke svakom od tri refleksiona servera prisutna u mreži, koristeći četvrtinu mrežnog kapaciteta kojeg imaju na raspolaganju. Svaki od ovih servera počinje da generiše saobraćaj koji je jednak polovini mrežnog kapaciteta bota koji je započeo napad. Ovim se efektivno postiže, ukoliko posmatramo sva tri refleksiona servera, ukupan faktor amplifikacije 6. Server mete napad vidi kao *HTTP GET*, a kao izvor ovog napada su refleksioni serveri.

3.6. OPIS KORIŠĆENIH KLASA HOSTOVA

- 1) *Bot* – ova klasa predstavlja klasu bota u okviru *botnet-a*.

Polja:

- *IPv4* (String) – predstavlja *IP* adresu bota.
- *bandwidth* (Integer) – predstavlja maksimalni mrežni protok linka bota u *Mbps*.
- *delay* (Integer) – predstavlja kašnjenje paketa do mete u ms.
- *Type* (Integer) – predstavlja tip uređaja (*1-desktop*, *2-laptop*, *3-ruter*, *4-web kamera*, *0-server*).
- *Status* (String) – predstavlja status bota u *botnet-u*, „*idle*“ ako bot ne napada nijednu metu, a u suprotnom ima string u odgovarajućem formatu.
- *currentBandwidth* (Integer) - predstavlja trenutni ukupni saobraćaj na linku posmatranog bota.
- *display* (PictureBox) – predstavlja sliku bota koja se prikazuje na radnoj površini, *WithEvents* označava da se koriste događaji kod ovog polja.
- *info* (Label) – ova labela stoji iznad slike bota i predstavlja identifikaciju bota u *botnet* mreži.
- *info2* (Label) – ova labela stoji desno od slike bota i predstavlja procentualno iskorišćenje mrežnog linka bota.
- *isVisible* (Boolean) – predstavlja flag da li se bot vidi na slici ili ne, koristi se interno u okviru metode iscrtavanja radne površine.
- *random* (Random) – predstavlja *Random* klasu koja je zajednička za sve botove i koristi se za generisanje *IP* adrese bota.
- *NetworkLog* (String) – koristi se za čuvanje mrežnog *log-a* napada datog bota.
- *oldXd* (Integer) – predstavlja staru X koordinatu slike bota i koristi se za izmenu pozicije slike bota.
- *oldYd* (Integer) – predstavlja staru Y koordinatu slike bota i koristi se za izmenu pozicije slike bota.
- *moving* (Boolean) – predstavlja flag koji se koristi kod crtanja – ukoliko se slika bota pomera trenutno onda se ne crta konekcija, radi preglednosti.

Metode:

- *GetIPv4* – predstavlja get metodu za *IP* adresu bota.

- *GetBandwidth* – predstavlja get metodu za maksimalni mrežni protok bota.
- *GetDelay* – predstavlja get metodu za mrežni *delay* bota.
- *GetTypeHost* – predstavlja get metodu za tip bota. Nazvana je ovako zato što je *GetType* metoda već zauzeta od strane okruženja.
- *Status* – vraća status bota za kog je metoda pozvana.
- *SetStatus* – postavlja status bota.
- *GetCurrentBandwidth* – vraća trenutni mrežni protok na linku.
- *SetCurrentBandwidth* – postavlja trenutni mrežni protok na linku.
- *SetLinkUsage* – vrši update info2 labele, kojom se postavlja procenat iskorišćenosti linka.
- *isGoodIP* – proverava da li je generisana *IP* adresa za bota ispravna (provera da li je adresa lokalna ili u opsegu nekorišćenih ili da li je već zauzeta).
- *generateIPv4* – predstavlja metodu koja generiše random *IPv4* adresu u obliku stringa uz pomoć random polja
- *New* – predstavlja konstruktor klase *Bot*, kojim se inicijalizuju sva polja, popunjavaju se početni sadržaji labela i postavljaju se parametri slike.
- *showImage* – prikazuje bota na radnoj površini.
- *hideImage* – briše bota sa radne površine.
- *Visible* – get metoda koja vraća da li je bot vidljiv.
- *drawConnection* – crta mrežni link koji vizuelno povezuje bota i internet.
- *Display_Click* – metoda koja omogućava prikazivanje detalja o botu na glavnom ekranu kad se klikne slika bota levim klikom, odnosno otvaranje menija sa različitim opcijama u zavisnosti od tipa bota ukoliko se klikne slika bota desnim klikom.
- *ResetLog* – metoda koja resetuje *NetworkLog* na prazan *string*.
- *AddLongEntry* – metoda koja dodaje zapis u mrežni *log* bota, zajedno sa vremenskim markerom.
- *GetNetworkLog* – get metoda koja vraća mrežni *log* bota.
- *OpenNetworkLog* – metoda koja otvara novu formu u kojoj se prikazuje mrežni *log*.
- *OpenTerminal* – metoda koja otvara novu formu u kojoj se prikazuje terminal za datog bota.

- *OpenGUI* – metoda koja otvara novu formu u kojoj se prikazuje *GUI* okruženje bota.
- *ShowNextGroup* – metoda koja služi da se prebaci prikaz sa trenutne grupe od podrazumevano 10 botova na sledeću grupu od 10 botova.
- *ShowPreviousGroup* – metoda koja služi da se prebaci prikaz sa trenutne grupe od podrazumevano 10 botova na prethodnu grupu od 10 botova.
- *Display_MouseDown* – metoda koja se poziva kada započinje pomeranje bota po ekranu od strane korisnika.
- *Display_MouseUp* – metoda koja se poziva kada se završi pomeranje bota po ekranu i onda se vrši postavljanje nove pozicije bota na radnoj površini.

2) *BotMaster* – predstavlja klasu uređaja napadača koji kontroliše *botnet*.

Polja:

- *IPv4* (String) – predstavlja IP adresu *botmaster-a*.
- *display* (PictureBox) – predstavlja sliku *botmaster-a* koja se prikazuje na radnoj površini, *WithEvents* označava da se koriste događaji kod ovog polja.
- *info* (Label) – ova labela stoji iznad slike *botmaster-a* i predstavlja identifikaciju *botmaster-a*.
- *random* (Random) – predstavlja Random klasu koja je zajednička za sve *botmaster-e* i koristi se za generisanje IP adrese *botmaster-a*.
- *oldXd* (Integer) – predstavlja staru X koordinatu slike *botmaster-a* i koristi se za izmenu pozicije slike *botmaster-a*.
- *oldYd* (Integer) – predstavlja staru Y koordinatu slike *botmaster-a* i koristi se za izmenu pozicije slike *botmaster-a*.
- *moving* (Boolean) – predstavlja flag koji se koristi kod crtanja – ukoliko se slika *botmaster-a* pomera trenutno onda se ne crta konekcija, radi preglednosti.

Metode:

- *GetIPv4* – predstavlja get metodu za IP adresu *botmaster-a*.
- *isGoodIP* – proverava da li je generisana IP adresa za *botmaster-a* ispravna (provera da li je adresa lokalna ili u opsegu nekorišćenih ili da li je već zauzeta).
- *generateIPv4* – predstavlja metodu koja generiše random IPv4 adresu u obliku string-a uz pomoć random polja
- *New* – predstavlja konstruktor klase *BotMaster*, kojim se inicijalizuju sva polja,

popunjavaju se početni sadržaji labela i postavljaju se parametri slike.

- *drawConnection* – crta mrežni link koji vizuelno povezuje *botmaster*-a i internet.
- *Display_Click* – metoda koja omogućava prikazivanje detalja o *botmaster*-u na main ekranu kad se klikne slika bota levim klikom, odnosno otvaranje menija sa različitim opcijama u zavisnosti od tipa bota ukoliko se klikne slika bota desnim klikom.
- *OpenGUI* – metoda koja otvara novu formu u kojoj se prikazuje *GUI* okruženje *botmaster*-a.
- *Display_MouseDown* – metoda koja se poziva kada započinje pomeranje *botmaster*-a po ekranu od strane korisnika.
- *Display_MouseUp* – metoda koja se poziva kada se završi pomeranje *botmaster*-a po ekranu i onda se vrši postavljanje nove pozicije *botmaster*-a na radnoj površini.
- *hideImage* – briše *botmaster*-a sa radne površine.

3) *Internet* – klasa kreirana da prezentuje internet na radnoj površini.

Polja:

- *display* (PictureBox) – predstavlja sliku interneta koja se prikazuje na radnoj površini, *WithEvents* označava da se koriste događaji kod ovog polja.
- *oldXd* (Integer) – predstavlja staru X koordinatu slike interneta i koristi se za izmenu pozicije.
- *oldYd* (Integer) – predstavlja staru Y koordinatu slike interneta i koristi se za izmenu pozicije.
- *moving* (Boolean) – predstavlja flag koji se koristi kod crtanja – ukoliko se slika interneta pomera trenutno onda se ne crta konekcija, radi preglednosti.

Metode:

- *New* – predstavlja konstruktor klase *Internet*, kojim se inicijalizuje slika interneta i postavlja na radnu površinu.
- *GetLocation* – predstavlja metodu koja vraća tačku trenutne lokacije slike interneta.
- *GetWidth* – predstavlja metodu koja vraća širinu slike interneta.
- *GetHeight* – predstavlja metodu koja vraća visinu slike interneta.

- *Display_MouseDown* – metoda koja se poziva kada započinje pomeranje interneta po ekranu od strane korisnika.
- *Display_MouseUp* – metoda koja se poziva kada se završi pomeranje interneta po ekranu i onda se vrši postavljanje nove pozicije interneta na radnoj površini.
- *hideImage* – briše internet sa radne površine.

4) *Server* – predstavlja klasu servera koji se koriste, može da bude refleksioni server, *botnet* server ili server koji se koristi kao meta napada.

Polja:

- *PortLimit* (Integer) – predstavlja maksimalan broj poluotvorenih konekcija, kao konstantu, koja je već podešena u sistemu. Koristi se kod simulacije SYN napada.
- *ACKWaitTime* (Integer) – predstavlja broj sekundi koji se čeka poslednji *ACK* segment u okviru *TCP* uspostavljanja konekcije. Koristi se kod simulacije SYN napada.
- *TCPACKQueue* (Queue Of Integer) – predstavlja red u kome se čuvaju svi zahtevi za uspostavljanje konekcije kod *TCP*-a, gde se čeka na posledni *ACK* segment od klijenta. Koristi se kod simulacije SYN napada.
- *PortSync* (Object) – koristi se kao objekat nad kojim se vrši sinhronizacija preko *SyncLock*.
- *attackSpeed* (Integer) – koristi se kao marker brzine pristizanja segmenata napadača kod SYN napada. Predstavlja ukupan broj segmenata za uspostavljanje konekcije koji se prime u sekundi u toku SYN napada.
- *thr1* (Thread) – predstavlja nit koja izvršava telo metode *ACKTimeoutThread*, koja reguliše broj poluotvorenih konekcija na osnovu brzine napada i maksimalnog vremena čekanja na *ACK* segment, u okviru strukture reda za čekanje sa tim zahtevima. Takođe reguliše i indikator koji se prikazuje na radnoj površini preko *SetLinkUsage* metode.
- *IPv4* (String) – predstavlja *IP* adresu servera, u ovom slučaju adresa se postavlja u okviru konstruktora.
- *bandwidth* (Integer) – predstavlja maksimalni mrežni protok servera izražen u *Mbps*.
- *currentBandwidth* (Integer) – predstavlja trenutni mrežni protok servera izražen u *Mbps*.

- *display* (PictureBox) – predstavlja sliku servera koja se prikazuje na radnoj površini, *WithEvents* označava da se koriste događaji kod ovog polja.
- *info* (Label) – ova labela stoji iznad slike servera i njegovu identifikaciju.
- *info2* (Label) – ova labela stoji desno od slike servera i predstavlja procentualno iskorišćenje mrežnog linka servera.
- *websiteServer* (Boolean) – flag koji označava da li server ima pokrenut *HTTP* protokol za hostovanje sajta.
- *websiteURL* (String) – predstavlja *URL* adresu sajta, ukoliko se sajt *HTTP* server.
- *website* (Site) – predstavlja sajt koji se hostuje na datom serveru.
- *openTCPPorts* (Queue Of Integer) – predstavlja red sa brojevima otvorenih portova na koje je moguće ostvariti konekciju preko *TCP* protokola.
- *oldXd*, *oldYd*, *oldXi*, *oldYi* (Integer) – varijable koje se koriste za promenu pozicije slike servera na ekranu.
- *moving* (Boolean) – predstavlja flag koji se koristi kod crtanja – ukoliko se interneta pomera trenutno onda se ne crta konekcija, radi preglednosti.
- *NetworkLog* (String) – predstavlja mrežni log svih napada koji su se dogodili na serveru, ukoliko je server meta, odnosno svih reflektovanih napada, ukoliko je dati server refleksioni server.

Metode:

- *GetIPv4* – metoda koja služi za dohvatanje *IP* adrese servera.
- *GetBandwidth* – metoda koja vraća maksimalni mrežni protok servera.
- *GetCurrentBandwidth* – metoda koja vraća trenutni mrežni protok na linku servera.
- *SetCurrentBandwidth* – metoda koja postavlja novu vrednost mrežnog protoka na serveru i poziva *SetLinkUsage* radi ažuriranja vrednosti iskorišćenosti linka.
- *AddOpenPort* – metoda koja dodaje otvoreni port na serveru.
- *GetOpenPorts* – metoda koja vraća red sa svim otvorenim *TCP* portovima na datom serveru.
- *New* – predstavlja konstruktor klase i u njemu se postavljaju vrednosti osnovnih polja, slika i labela na osnovu imena servera.
- *SetLinkUsage* – vrši update *info2* labele, kojom se postavlja procenat

iskorišćenosti linka.

- *drawConnection* – crta mrežni link koji vizuelno povezuje server i internet.
- *Display_MouseDown* – metoda koja se poziva kada započinje pomeranje servera po ekranu od strane korisnika.
- *Display_MouseUp* – metoda koja se poziva kada se završi pomeranje servera po ekranu i onda se vrši postavljanje nove pozicije servera na radnoj površini.
- *Display_Click* – metoda koja se poziva kad se detektuje klik na slici servera i tad se prikazuje odgovarajući meni ukoliko je klik desni, odnosno prikazuju se detalji servera na radnoj površini ukoliko je levi klik.
- *hideImage* – metoda koja briše sliku i labele vezane za server sa radne površine.
- *isWebServer* – get metoda koja vraća da li je server *web* tipa, odnosno da li hostuje neki sajt.
- *getWebsiteURL* – metoda koja vraća *URL* od *web* sajta koji server hostuje.
- *getWebsite* – metoda koja vraća *web* sajt koji je hostovan na serveru.
- *openWebsite* – metoda za zadatu adresu traži odgovarajući sajt i ukoliko je server koji ga hostuje u funkciji i nije napadnut vraća taj sajt a u suprotnom vraća sajt greške.
- *resolveURLtoIP* – ova metoda vraća *IP* adresu servera na kom je hostovan dati sajt.
- *ResetLog* – predstavlja metodu koja resetuje mrežni *log* o napadima servera na prazan string.
- *AddLogEntry* – dodaje zapis u mrežnom *log*-u servera, zajedno sa vremenskim markerom.
- *GetNetworkLog* – get metoda koja vraća mrežni *log* servera.
- *OpenNetworkLog* – metoda koja otvara novu formu gde se prikazuje mrežni *log* servera.
- *GetSYNAttackSpeed* – metoda koja vraća brzinu *SYN* napada kao broj requestova po sekundi.
- *KillThread* – metoda koja ubija nit servera koja je zadužena za *SYN* napade.

4. NAČIN KORIŠĆENJA APLIKACIJE

U okviru ovog poglavlja biće opisani preduslovi za pokretanje aplikacije, kao i dato uputstvo za njeno korišćenje, uz par primera, kojima se pokazuje konceptualno prikaz više napada i njihov uticaj na metu.

4.1. PREDUSLOV ZA POKRETANJE APLIKACIJE

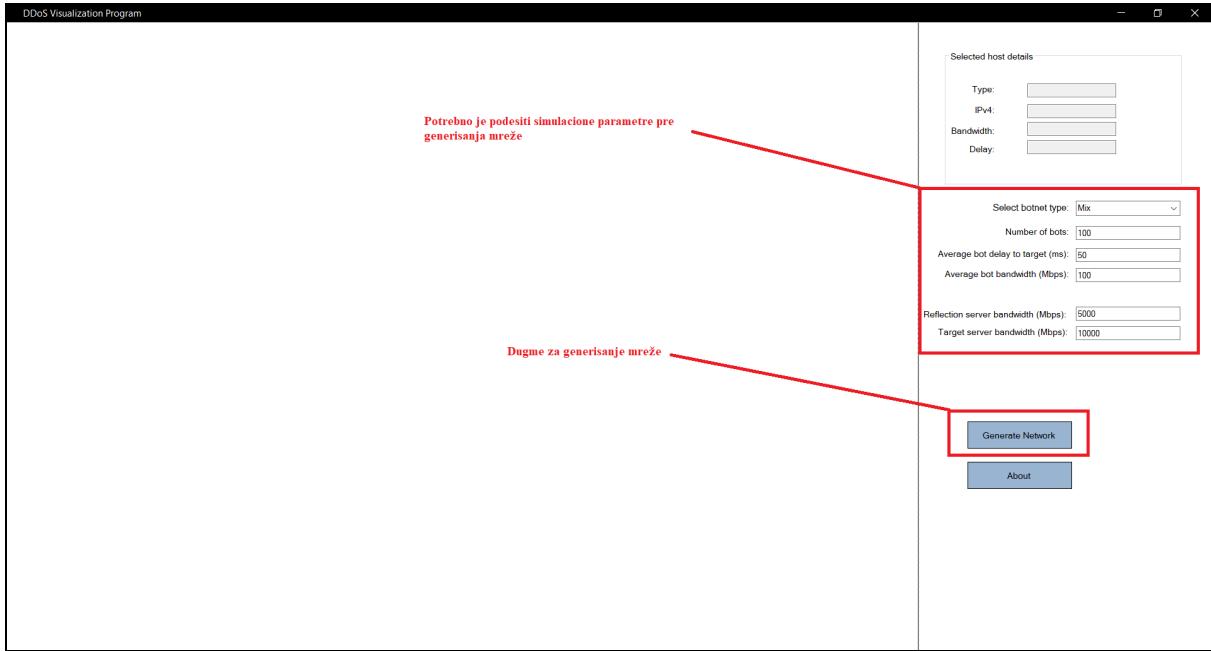
Budući da je aplikacija realizovana koristeći *.NET Framework 4.5* [34] kao implementaciono okruženje, potrebno je skinuti framework sa *Microsoft*-ovog sajta [33] i instalirati, da bi aplikacija mogla da funkcioniše.

4.2. OSNOVNI PROZOR APLIKACIJE I GENERISANJE MREŽE

Pokretanjem aplikacije se otvara glavni prozor (slika 4.1). U okviru desnog dela glavnog prozora se nalaze postavke za kreiranje mreže uređaja koja će biti prikazana:

- Tip *botnet*-a (padajući meni *Select botnet type*): *Botnet* u okviru ove aplikacije je realizovan kao *HTTP botnet*, gde *botnet* server sa botovima komunicira preko *HTTP*, a istovremeno preko *HTTP* stranice *botnet*-a se kontrolišu botovi i započinju napadi. Ovo podešavanje se odnosi na tip uređaja koji se nalazi u *botnet*-u, da li su to *IoT* uređaji (ruteri ili *IP* kamere), računari (serveri, desktop i laptop uređaji) ili mix obe vrste uređaja.
- Broj botova (polje za upis teksta *Nubmber of bots*): U ovo polje se unosi broj botova u okviru generisanog *botnet*-a.
- Prosečan *delay* od bota do servera mete (polje za upis teksta *Average bot delay to target (ms)*): Ovde se unosi prosečna vrednost vremena potrebnog da paketi stignu od bota do servera mete, koja se koristi kod *ping*-ovanja, kod terminalnih uređaja i ima informacioni karakter. Vrednost se generiše kao uniformna u skupu diskretnih vrednosti za *Integer* u granicama od [1,x] milisekundi, gde je x broj koji je korisnik zadao.
- Prosečan mrežni kapacitet bota (polje za upis teksta *Average bot bandwidth (Mbps)*): Ovde se unosi prosečna vrednost mrežnog kapaciteta svakog bota, koja se koristi kod *DDoS* napada. Vrednost se generiše kao uniformna u skupu diskretnih vrednosti za *Integer* u granicama od [1,x] Mbps, gde je x broj koji je korisnik zadao.

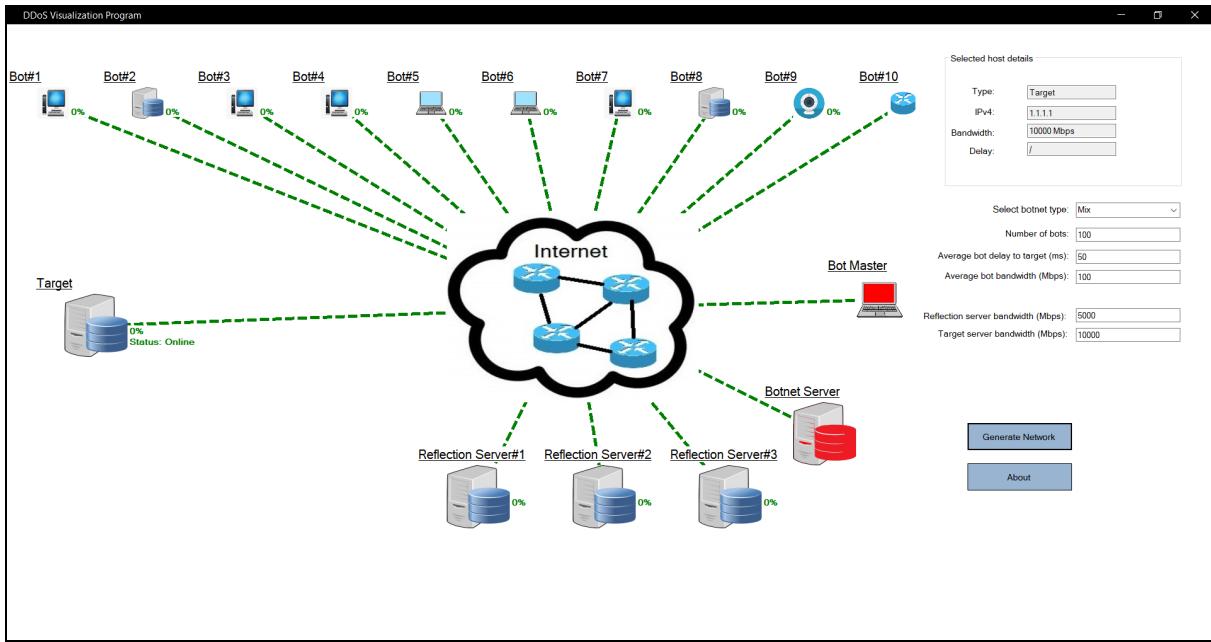
- Mrežni kapacitet refleksionih servera (polje za upis teksta *Reflection server bandwidth (Mbps)*): U okviru ovog polja se upisuje vrednost mrežnog kapaciteta refleksionih servera u *Mbps*, koji se koristi za *XMLRPC* tip napada.
- Mrežni kapacitet servera mete (polje za upis teksta *Target server bandwidth(Mbps)*): U okviru ovog polja se upisuje vrednost mrežnog kapaciteta servera mete. Koristi se kod svih *DDoS* napada.



Slika 4.1 – Glavni prozor aplikacije pre generisanja mreže

Pritiskom na dugme za generisanje mreže (dugme *Generate Network*) dobija se nov izgled u levom delu aplikacije, kao na slici 4.2. Ukoliko se unesu pogrešni parametri (broj botova manji od 0, *delay* manji od 1 ms, mrežni kapaciteti manji od 1 *Mbps*), dobija se odgovarajuća poruka o grešci i nova mreža se ne generiše.

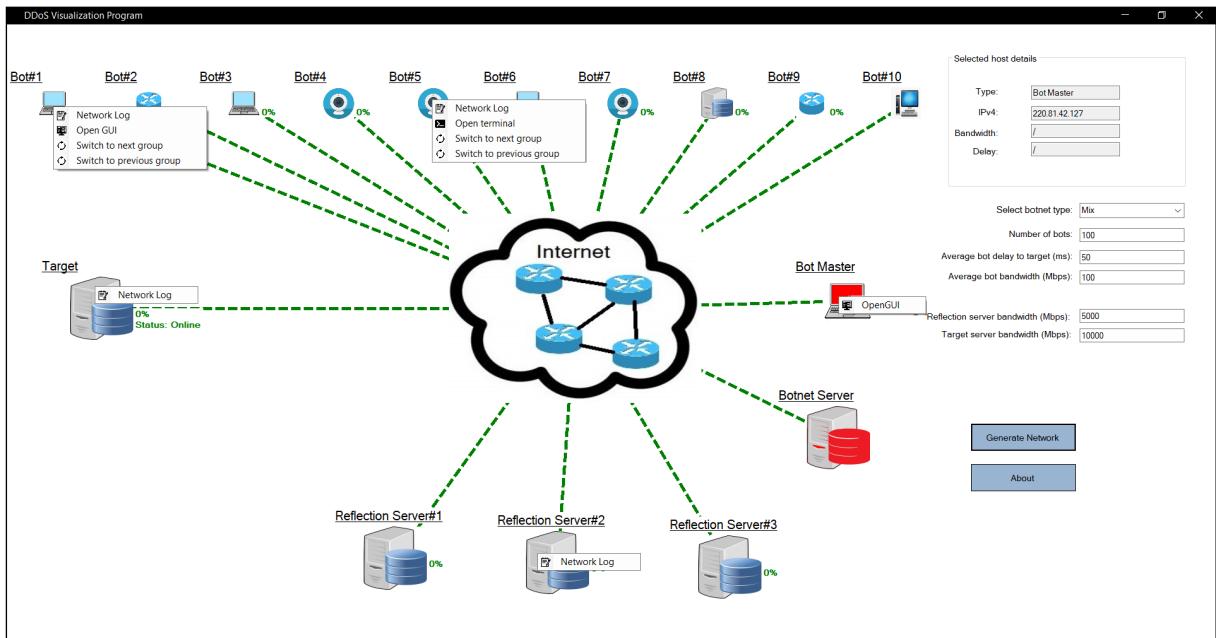
Ukoliko korisnik želi da napravi novu simulaciju, sa novim parametrima, u bilo kom trenutku je moguće uneti nove parametre simulacije i pokrenuti je ponovo klikom na odgovarajuće dugme. IP adrese generisanih botova su nasumične, dok su adrese *target* servera, refleksionih servera i *botnet* servera iste svaki put kad se generiše nova mreža.



Slika 4.2 – Prikaz generisane mreže u okviru aplikacije

Korisnik sad može da vrši vizuelne izmene prikazane mreže, pomeranjem botova, servera i interneta, prevlačenjem svakog *host-a* na njegovu novu poziciju, koristeći levi klik miša. U okviru *Selected host details* prozora mogu se videti detalji vezani za selektovanog *host-a*, koristeći levi klik miša.

Desnim klikom na sliku svakog *host-a* izlazi odgovarajući meni, preko koga je moguće otvoriti nove forme i nadalje menjati prikaz mreže sa slike. Na slici 4.3 prikazane su različite opcije menija, u zavisnosti od tipa *host-a* koji je selektovan.



Slika 4.3 – Prikaz menija za različite tipove host-ova

Funkcije odgovarajućih opcija u okviru menija date su u nastavku:

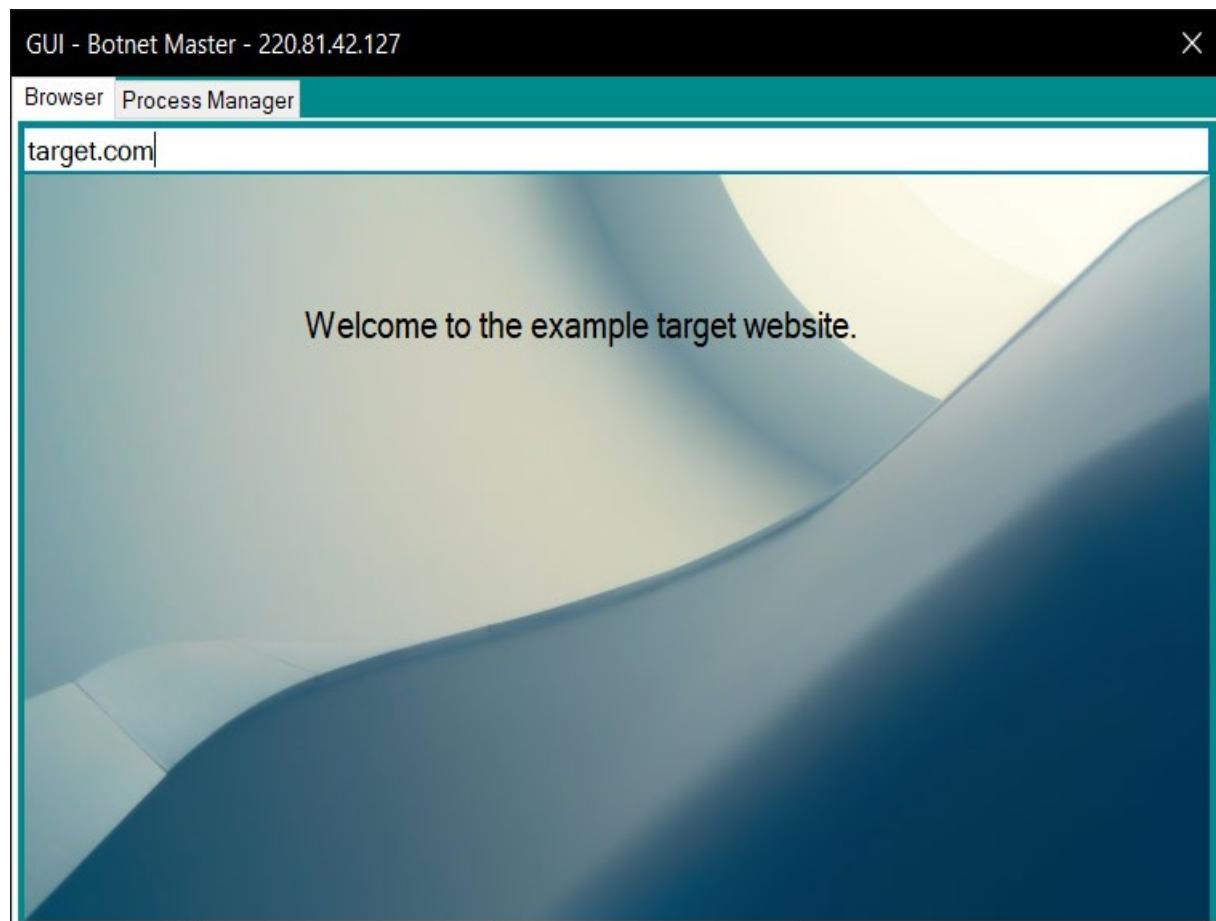
- *Switch to next group*: Ova opcija prebacuje prikaz na grupu od sledećih 10 botova iz mreže. Ukoliko se dođe do poslednje grupe i ponovo pritisne ova opcija, prikazaće se grupa od prvih 10 botova.
- *Switch to previous group*: Ova opcija prebacuje prikaz na grupu od prethodnih 10 botova iza mreže. Ukoliko se dođe do prve grupe i ponovo pritisne ova opcija, prikazaće se grupa od poslednjih 10 botova.
- *NetworkLog*: Ova opcija otvara novu formu u kojoj se nalaze mrežni *log-ovi* za odgovarajućeg *host-a*. Za server mete (*Target server*) ovde se čuvaju informacije o detektovanim *DDoS* napadima (tip napada, izvor iz ugla servera mete i vremenski marker) – kad su napadi počeli i završili se. Za refleksione servere (*Reflection Server#1*, *Reflection Server#2*, *Reflection Server#3*) čuvaju se informacije o tome kad su započeti i završeni napadi koji se reflektuju uz pomoć njih. Za botove se čuvaju informacije o tome kad su započeti i završeni napadi.
- *Open GUI*: U okviru ove opcije se otvara nova forma koja predstavlja *GUI* okruženje, u okviru koga postoje dva prozora – *Browser* i *Process Manager*. U okviru *Browser* prozora moguće je otvoriti jedan od dva postojeća *web* sajta koristeći njihovu *URL* ili *IP* adresu (*target.com* ili *1.1.1.1* za sajt mete i *botnet.com* ili *2.2.2.2* za sajt za kontrolu *botnet-a*). U okviru *Process Manager* prikazuje se nasumično generisana lista procesa. Ukoliko je posmatrani *host* bot, postojaće stavka malicioznog programa, koji podseća korisnika da se kontrola nad botom i njegova povezanost sa ostatkom *botnet-a* ostvarena koristeći dati maliciozni program. Ova stavka postoji samo za računare i servere u okviru *botnet-a*, kao i za računar *botmaster-a*, koji kontroliše *botnet*.
- *Open Terminal*: Ova stavka otvara novu formu u kojoj se prikazuje okruženje terminala za botove koji su *IP* kamere ili ruteri. Spisak dostupnih komandi se može videti kucanjem ? ili *commands*. Moguće je *ping*-ovanje bilo kog *host-a* u okviru generisane mreže. Ukoliko se *ping*-uje drugi bot, ili neki od servera koji nisu serveri mete, *time* će biti random generisana vrednost, dok ukoliko se *ping*-uje server mete *time* će biti *delay* koji je generisan u okviru granica koje je korisnik predvideo. Takođe moguće je prikazati sve procese koji se izvršavaju na uređaju, koji su takođe nasumično generisani.

4.3. DDOS NAPADI, POKRETANJE I ZAUSTAVLJANJE NAPADA

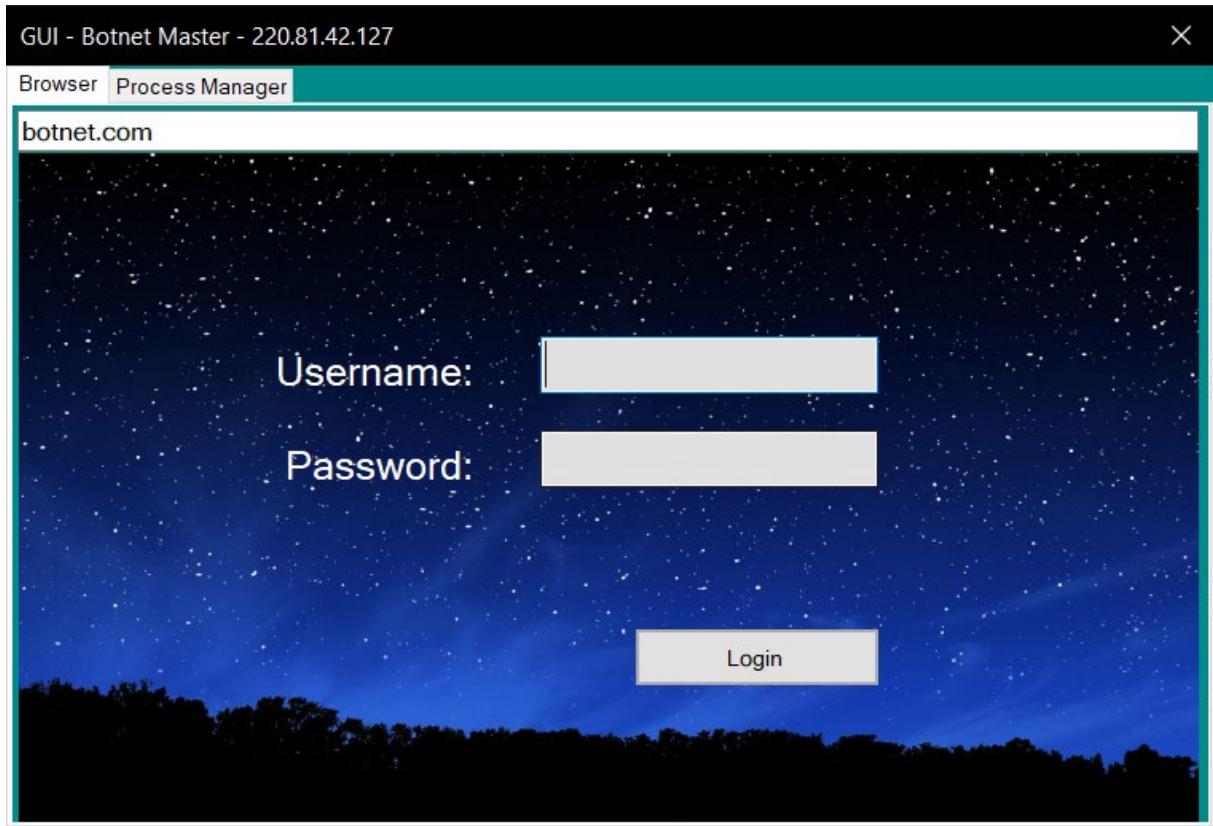
Dakle, otvaranje sajtova je moguće koristeći *Browser* u okviru *GUI* interfejsa. Dva sajta koja postoje u mreži su sajt mete napada (*domen target.com* ili *IP* adresa **1.1.1.1**) i sajt za kontrolu *botnet-a* (*domen botnet.com* ili *IP* adresa **2.2.2.2**).

Sajt mete je prikazan na slici 4.4. Za otvaranje sajtova je predviđeno korišćenje *botmaster-a*, koji proverava da li je sajt mete (servis koji on hoće da napadne) funkcionalan i koji kontroliše *botnet*, pokretanjem i zaustavljanjem napada. Otvaranje sajtova je moguće i korišćenjem botova koji imaju *GUI* interfejs, ukoliko se oni ne koriste za trenutno izvođenje napada.

Sajt za kontrolu *botnet-a* je prikazan na slici 4.5. Naravno, radi pristupanja interfejsu za kontrolu *botnet-a*, potrebno je da se obavi autentikacija, korisničkim imenom **admin** i lozinkom **123**, koja postoji kao obavezna stavka kod svakog *HTTP* *botnet-a*, radi onemogućavanja neovlašćenima da pristupe i kontrolišu *botnet*.



Slika 4.4 – Prikaz sajta mete u slučaju kad nema DDoS napada



Slika 4.5 – Prikaz forme za logovanje botnet sajta

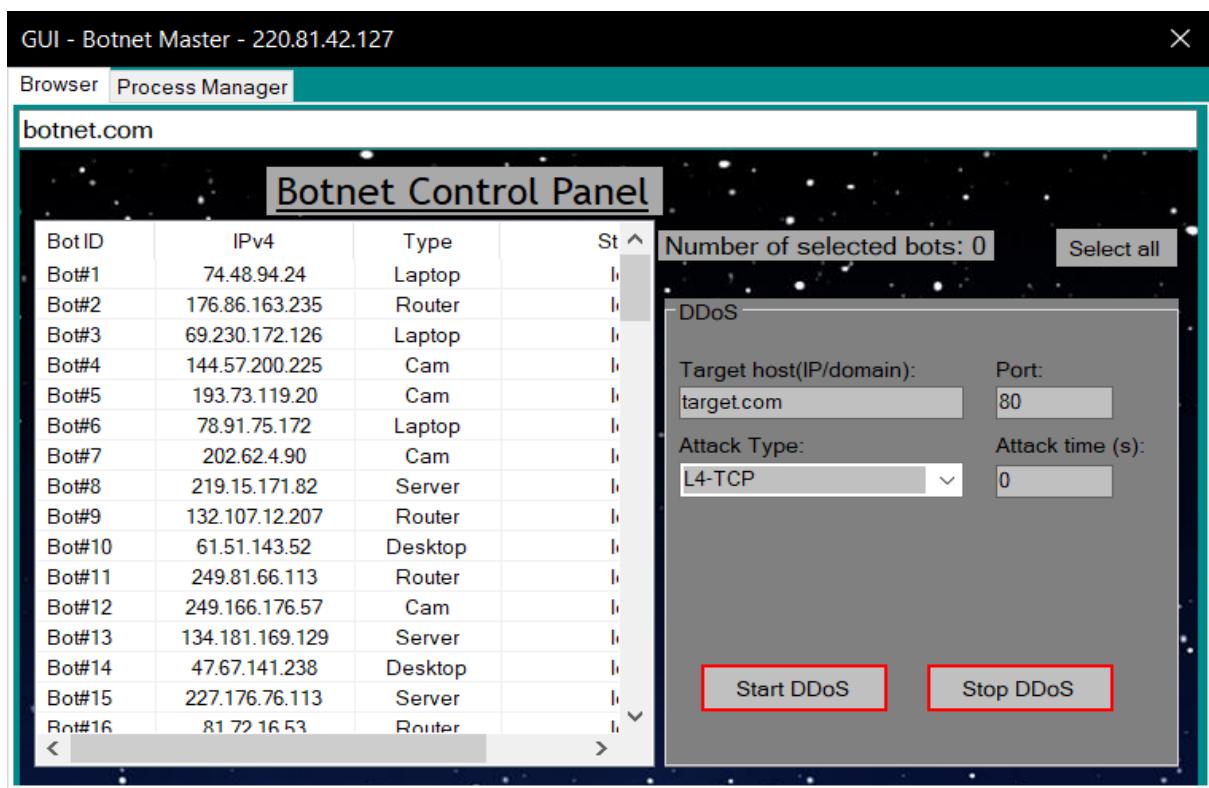
Nakon procesa logovanja, otvara se panel u okviru koga je prikazana lista botova, njihov tip i IP adresa i status koji govori da li se trenutno dešava neki napad ili ne (status „Idle“ označava da bot trenutno ne napada nijednu metu). Panel za kontrolu botnet-a je prikazan na slici 4.6.

Pokretanje napada se vrši selektovanjem određenog broja botova ili koristeći dugme *Select all* (selektovanje više botova istovremeno se radi držeći taster *CTRL* i koristeći levi klik na svakom botu koji treba da se selektuje). Broj selektovanih botova je moguće videti u svakom momentu u okviru labela *Number of selected bots*.

Nakon procesa selekcije botova koji vrše napad, potrebno je izabrati tip napada i popuniti odgovarajuća tekstualna polja o meti:

- IP adresu ili domen mete *DDoS* napada.
- Port mete koji se koristi u napadu.
- Vreme napada u sekundama (ukoliko se navede 0 sekundi napad traje dok ga *botmaster* ne prekine).

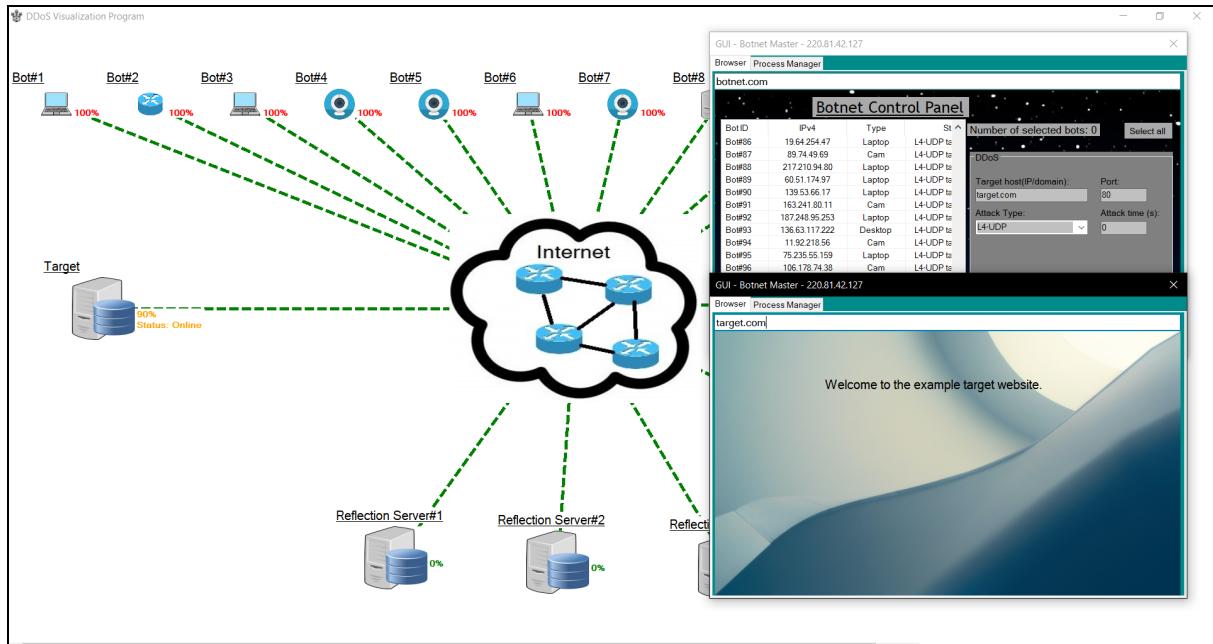
Više informacija o napadima se nalazi u delu rada koji se bavi implementacijom aplikacije. Sada je moguće započeti napad klikom na dugme *Start DDoS*. Napad je takođe moguće zaustaviti u bilo kom momentu, koristeći dugme *Stop DDoS*, koje prekida napad za sve selektovane botove. Naravno, aplikacija podržava više različitih tipova napada istovremeno.



Slika 4.6 – Prikaz kontrolnog panela botneta

4.4. PRIMERI DDOS NAPADA

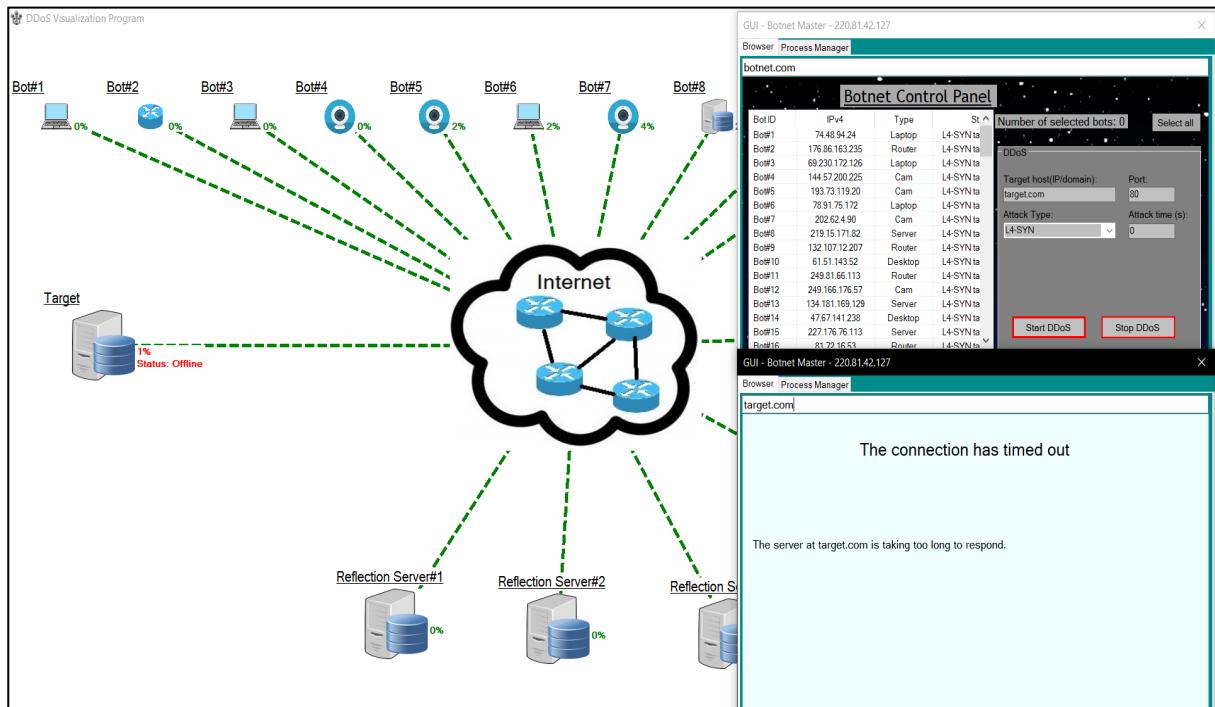
Primer započetog napada, metodom *UDP*, prikazan je na slici 4.7. Koriste se podrazumevani parametri mreže koji se dobijaju svaki put kad se pokrene program. Kao što se vidi, indikatorom kod servera mete, celokupni napad je dostigao ukupan kapacitet od 90% mrežnog kapaciteta sa kojim meta raspolaže. Na osnovu toga server mete je i dalje *online* i moguće je pristupiti sajtu target.com od strane *botmaster-a*. Ako bi probali da otvorimo sajt koristeći *GUI* interfejs botova, ovo ne bi uspelo, budući da botovi nemaju na raspolaganju dovoljno mrežnog kapaciteta za otvaranje sajta.



Slika 4.7 – Primer UDP napada koji nije uspeo da obori metu

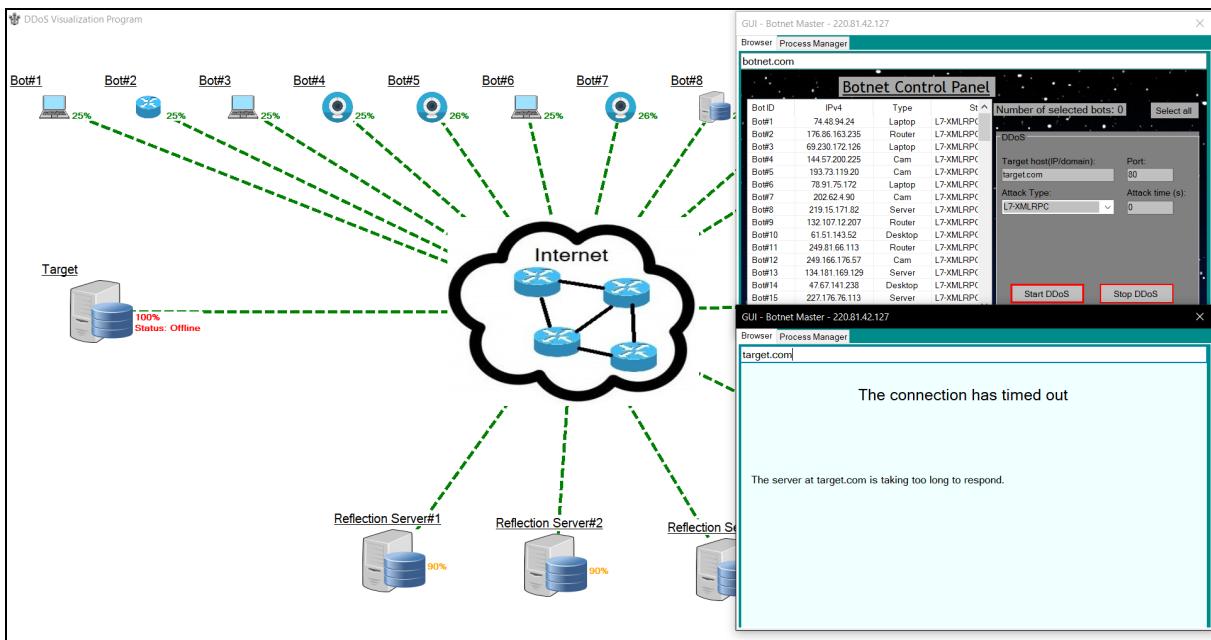
UDP napad, kao što se može videti, nije uspeo da obori server mete. Moguće je i dalje pristupiti sajtu target.com i indikator na radnoj površini je žute boje. Međutim ukoliko bi probali da napadnemo metu koristeći SYN napad, ne menjajući konfiguracione parametre, nakon što prekinemo trenutni UDP napad, videli bi da meta odlazi *offline* kao na slici 4.8.

Ovo je posledica prirode SYN napada, budući da je napad sad bio dovoljan da iskoristi sve resurse kod servera mete i na taj način je onemogućio dodatne konekcije ka meti, iako je SYN napad generisao vrlo malo saobraćaja.



Slika 4.8 – Primer SYN napada koji je uspešno oborio server mete

Kao primer još jednog napada koji bi uspešno oborio sajt mete, koristićemo *XMLRPC* napad. Budući da botovi nemaju dovoljan mrežni kapacitet, iskoristiće metodu amplifikacije i refleksije napada, uz pomoć refleksionih servera. Ova metoda će, kao i *SYN* napad uspeti da obori server mete i on više neće biti funkcionalan (slika 4.9).



Slika 4.9 – Primer XMLRPC napada koji je uspešno oborio server mete

5. ZAKLJUČAK

DDoS napadi će nastaviti da predstavljaju jednu od najvećih opasnosti kod funkcionalisanja interneta i servisa koji ga koriste. Tema ovog rada je bilo upoznavanje sa konceptom *DDoS* napada i principom njegovog funkcionalisanja kroz implementaciju interaktivne aplikacije u kojoj bi korisnik imao vizuelni pregled cele mreže, mogućnost pokretanja jednog od više različitih tipova napada, kao i pregled posledica ovakvih napada.

Budući da aplikacija predstavlja pojednostavljeni prikaz mreže i relativno jednostavnu implementaciju *DDoS* napada, jedan od vidova unapređenja aplikacije bio bi korišćenje kompleksnijih struktura podataka i koncepta multiprogramiranja na višem nivou uz analizu protokola koji se koriste za izvođenje napada i njihov uticaj na stvarne sisteme, tako da se dobije preciznija simulacija *DDoS* napada. Takođe, kao jedno od mogućih unapređenja aplikacije je dodavanje različitih vidova odbrane od *DDoS* napada, koji bi u određenoj meri sprecili ili smanjili njihov uticaj.

LITERATURA

1. Nikhil Tripathi, B.M. Mehtre, *DoS and DDoS Attacks: Impact, Analysis and Countermeasures*, December 2013.
2. Vivek Ganti, Omer Yoachimik, *DDoS attack trends for 2021 Q2* <https://blog.cloudflare.com/ddos-attack-trends-for-2021-q2/> (pristupljeno avgusta 2021.)
3. *Digital Attack Map* <https://www.digitalattackmap.com/> (pristupljeno avgusta 2021.)
4. Piyush Saxena, *OSI Reference Model – A Seven Layered Architecture of OSI Model*, November 2014.
5. *List of network protocols (OSI model)*
[https://en.wikipedia.org/wiki/List_of_network_protocols_\(OSI_model\)/](https://en.wikipedia.org/wiki/List_of_network_protocols_(OSI_model)/) (pristupljeno avgusta 2021.)
6. K. Munivara Prasad, A. Rama Mohan Reddy, K. Venugopal Rao, *DoS and DDoS Attacks: Defense, Detection and Traceback Mechanisms -A Survey*, **Global Journal of Computer Science and Technology**, 2014.
7. Gregory Fedynyshyn, Mooi Choo Chuah, and Gang Tan, *Detection and Classification of Different Botnet C&C Channels*, September 2011.
8. *Mirai malware* [https://en.wikipedia.org/wiki/Mirai_\(malware\)](https://en.wikipedia.org/wiki/Mirai_(malware)) (pristupljeno avgusta 2021.)
9. Mohammad Karami, Damon McCoy, *Understanding the Emerging Threat of DDoS-As-a-Service*, **George Mason University**, August 12. 2013.
10. Krushang Sonar1, Hardik Upadhyay, *A Survey: DDOS Attack on Internet of Things*, **International Journal of Engineering Research and Development**, November 2014.
11. Fu-Hau Hsu, Yan-Ling Hwang, Cheng-Yu Tsai, Wei-Tai Cai, Chia-Hao Lee and KaiWei Chang, *TRAP: A Three-Way Handshake Server for TCP Connection Establishment*, **Applied Sciences Journal**, November 16 2016.
12. *Cloudflare Learning Center – DDoS attacks* <https://www.cloudflare.com/en-gb/learning> (pristupljeno avgusta 2021.)
13. Ghafar A. Jaafar, Shahidan M. Abdullah, and Saifuladli Ismail, *Review of Recent Detection Methods for HTTP DDoS Attack*, **Journal of Computer Networks and Communications**, January 10 2019.

14. Wordpress <https://wordpress.org/> (pristupljeno avgusta 2021.)
15. Kevin W., *What Is Xmlrpc.php in WordPress and Why You Should Disable It*, <https://www.hostinger.com/tutorials/xmlrpc-wordpress> (pristupljeno avgusta 2021.)
16. Richard B., *What is a Pingback in WordPress?*, <https://www.hostinger.com/tutorials/what-is-pingback> (pristupljeno avgusta 2021.)
17. Kamal Alieyan, Shafiq Ul Rehman, Mohammed Anbar, *An overview of DDoS attacks based on DNS*, October 2016.
18. L. Rudman, B. Irwin, *Characterization and Analysis of NTP Amplification Based DDoS Attacks*, 2015.
19. RFC 864 <https://datatracker.ietf.org/doc/html/rfc864>
20. Oliver Adam, *Deep inside CHARGEN flood attacks*, August 2018., <https://www.link11.com/en/blog/threat-landscape/deep-inside-chargen-flood-attacks/>
21. RFC 791 <https://datatracker.ietf.org/doc/html/rfc791>
22. *Ping of death* https://en.wikipedia.org/wiki/Ping_of_death (pristupljeno septembra 2021.)
23. *Smurf attack* https://en.wikipedia.org/wiki/Smurf_attack (pristupljeno septembra 2021.)
24. *Visual Basic .NET* <https://docs.microsoft.com/en-us/dotnet/visual-basic/> (pristupljeno septembra 2021.)
25. GNS3 <https://www.gns3.com/> (pristupljeno septembra 2021.)
26. *Cisco Packet Tracer* <https://www.netacad.com/courses/packet-tracer> (pristupljeno septembra 2021.)
27. RFC 792 <https://datatracker.ietf.org/doc/html/rfc792>
28. RFC 2068 <https://datatracker.ietf.org/doc/html/rfc2068>
29. *Low Orbit Ion Cannon* https://en.wikipedia.org/wiki/Low_Orbit_Ion_Cannon (pristupljeno septembra 2021.)
30. *High Orbit Ion Cannon* https://en.wikipedia.org/wiki/High_Orbit_Ion_Cannon (pristupljeno septembra 2021.)
31. *Slowloris* [https://en.wikipedia.org/wiki/Slowloris_\(computer_security\)](https://en.wikipedia.org/wiki/Slowloris_(computer_security)) (pristupljeno septembra 2021.)
32. *Microsoft Visual Studio* <https://visualstudio.microsoft.com/> (pristupljeno septembra 2021.)

33. Microsoft .NET Framework version 4.5 <https://www.microsoft.com/en-us/download/details.aspx?id=30653> (pristupljeno septembra 2021.)

34. Microsoft Introduction to .NET <https://docs.microsoft.com/en-us/dotnet/core/introduction> (pristupljeno septembra 2021.)

PRILOG A

U ovom poglavlju su dati detalji vezani za *OSI* referentni model, primer jednog od najpoznatijih *botnet* sistema i opis *DDoS-for-hire* sistema.

OSI REFERENTNI MODEL

OSI (*Open System Interconnection*) predstavlja referentni model koji je predložila *ISO* (*International Standard Organization*) 1983. godine, koji pokriva sve aspekte mrežne komunikacije. Svrha nastajanja *OSI* referentnog modela je da se organizuje i pojednostavi proces dizajniranja i razumevanja mrežnih protokola, preko uvođenja koncepta slojeva (ukupno ih ima 7), gde svaki sloj može da komunicira samo sa svojim susednim slojevima i gde viši slojevi enkapsuliraju niže, dok se pravi prenos podataka od jednog do drugog *host-a* obavlja samo u okviru prvog (fizičkog sloja) [4].

OSI slojevi	Protokoli
7. Sloj aplikacije	HTTP, HTTPS, POP3, IMAP, SMTP, IRC, SSH, NFS, DNS
6. Sloj prezentacije	SSL, TLS
5. Sloj sesije	SMB, RPC, SOCKS
4. Transportni sloj	TCP, UDP, QUIC
3. Mrežni sloj	IPv4, IPv6, ARP, ICMP, RIP, OSPF, IPsec
2. Sloj veze	Ethernet, IEEE 802.11 WiFi, VLAN, STP
1. Fizički sloj	RS232, Ethernet standardi (100BASE-T, 1000BASE-T...), CAN, DSL, USB

Slika A.1 – Slojevi i karakteristični protokoli OSI referentnog modela [5]

Struktura *OSI* referentnog modela je prikazana na slici A.1, kao i karakteristični protokoli za svaki sloj. Neki od ovih protokola se logički nalaze na više slojeva.

Funkcije slojeva *OSI* referentnog modela su date u nastavku: [4]

1. *Fizički sloj*: Ovaj sloj je nadležan za prenos bajtova podataka, od jednog do drugog *host-a* u okviru specifičnog medijuma za prenos. U okviru fizičkog sloja su standardizovane mehaničke i električne specifikacije medijuma za prenos,

enkodovanje bitova, njihovo trajanje i sinhronizacija.

2. *Sloj veze*: Ovaj sloj je nadležan za organizaciju bajtova podataka u logičke grupe podataka koje se zovu frejmovi, za detekciju greške pri prenosu frejmova, kontrolu protoka i brzine slanja frejmova. Kao način adresiranja koristi *MAC (Media Access Controller)* adrese koje predstavljaju lokalne hardverske adrese *host-ova* u okviru jedne mreže. *MAC* adrese su dužine 6 bajtova i uglavnom se predstavljaju u heksadecimalnoj formi sa dvotačkama ili crticama između svakog bajta.
3. *Mrežni sloj*: Mrežni sloj kao osnovnu jedinicu podataka koristi pakete i on je odgovoran za njihovo rutiranje i adresiranje (preko *IP* adresa). Dakle na ovom sloju rade i protokoli internog rutiranja koji služe da odrede rutu u mreži kojom će da ide paket.
4. *Transportni sloj*: Transportni sloj je zadužen za prenos jedne poruke kao celine između više *host-ova*, upravljanjem brzine slanja i kontrolom prenosa svakog dela poruke. Jedinica podataka na ovom sloju je segment za *TCP* i datagram za *UDP*. Kao dva bitna protokola u okviru transportnog sloja javljaju se *TCP* i *UDP*. *TCP (Transmission Control Protocol)* predstavlja konepciono orijentisani protokol, koji koristi koncept *handshake-a* između dva *host-a* za uspostavljanje konekcije i za potvrdu da je poslat paket pristigao do odredišta i ukoliko nije traži se retransmisija. *TCP* takođe vodi računa o redosledu pristizanja paketa. *UDP (User Datagram Protocol)* je sušta suprotnost *TCP* protokolu, on radi na „*best effort*“ principu i predstavlja *connectionless* protokol, tjst. nije potreban koncept *handshake-a* i prethodnog uspostavljanja konekcije. On ne proverava da li su poslati paketi pristigli i da li su stigli u korektnom redosledu. Prednost *UDP* prenosa je njegova brzina.
5. *Sloj sesije*: Ovaj sloj omogućava komunikaciju između dva *host-a* u obliku sesije – aplikacije na oba *host-a* mogu da šalju i primaju poruke sve dok je sesija uspostavljena. Ovaj sloj se bavi uspostavljanjem i raskidanjem sesije, slanjem podataka i sigurnosnih informacija vezanih za učesnike u sesiji.
6. *Sloj prezentacije*: Sloj prezentacije se bavi prikazom podataka koji se šalju. U okviru ovog sloja obavlja se enkripcija podataka, kompresija teksta i slike i reformatiranje. Enkripcija se koristi radi autentikacije i zaštite privatnosti, dok kompresija dovodi do smanjenja količine bajtova koja se šalje. Ovaj sloj je nadležan da konvertuje podatke u onaj tip koji korisnička aplikacija koristi.
7. *Sloj aplikacije*: U okviru ovog sloja se nalaze protokoli aplikacija koje korisnik na računaru koristi i omogućavaju interfejs i podršku za mnoge servise kao što su elektronska pošta, upravljanje bazama podataka, slanje fajlova, prikaz web stranica itd.

MIRAI BOTNET

Jedan od najpoznatijih *botnet* sistema koji se koristio za *DDoS* napade je *Mirai botnet*, koji je zloupotrebljavao jednostavne *Linux* uređaje kao što su *IP* kamere ili ruteri. Svaki od tih uređaja je skenirao internet tražeći nove uređaje koje bi zarazio i uveo u *botnet*, koristeći se tabelom sa više od 60 podrazumevanih (fabričkih) kombinacija kredencijala za logovanje putem *Telnet* protokola (7. nivo *OSI* modela), koje korisnici *IoT* uređaja nisu često menjali. Zaraženi uređaji nastavljaju da funkcionišu, ali generišu veću količinu mrežnog saobraćaja. Pre nego što su uhapšeni, autori su objavili kod od ovog malicioznog programa, na osnovu koga su kasnije nastale druge varijante ovog programa, koje nastavljaju da budu aktivne. [8]

Mirai je korišćen za napad na francuskog hosting provajdera OVH, gde je 1 *Tbps* ukupna zabeležena brzina napada. Takođe ovaj *botnet* je odgovoran za napad 21. oktobra 2016. godine na mrežu *DNS* provajdera DYN, sa detektovanim ukupnom brzinom od oko 1.2 *Tbps*, što je prouzrokovalo probleme korisnika pri pristupanju značajnom broju sajtova velikih kompanija kao što su GitHub, Twitter, Reddit, Netflix, Airbnb itd. Ovaj napad će ostati upamćen kao jedan od najvećih u istoriji interneta. [8]

DDOS KAO SERVIS

Kao posledica neefikasnosti besplatnih alata, koji omogućavaju napadačima sa manje iskustva i znanja da pokrenu napade, koji su pomenuti u poglavlju 2.3., nastali su tzv. *DDoS-for-hire* (ili *booter*) servisi, gde za relativno malu količinu novca je moguće kupiti pristup sistemu koji služi za pokretanje *DDoS* napada. Ovi servisi se najčešće reklamiraju kao *stress-testing* servisi i kao njihova komercijalna namena se navodi testiranje rada sistema protiv *DDoS* napada. Struktura koja stoji iza ovakvog servisa je najčešće *botnet* ili mreža servera sa velikim mrežnim kapacitetom.

U početku ovakvi servisi su korišćeni od strane igrača *online* računarskih igara, radi napadanja protivnika (čime bi protivnik dobio diskvalifikaciju), dok danas se uglavnom koriste protiv *web* sajtova srednje veličine, imajući u vidu poboljšanja mrežne sigurnosti u okviru video igara.

Kao jedan od primera *DDoS-for-hire* servisa u okviru [9] se navodi *TwBooter*, koji je za funkcionisanje koristio infrastrukturu sa ukupno 15 različitih servera sa kojih su pokretani napadi. Postojala je opcija biranja jednog od 12 različitih napada, kao što su *SYN*, *UDP*, *HTTP GET*, *HTTP POST*, *HTTP HEAD*, *RUDY* i aplikaciono specifični napadi kao što su *Slowloris* za *Apache web* servere. Za prikrivanje izvora napada korišćeni su *spoof-ovanje source IP* dela paketa ili korišćenje velikog broja *proxy* servera.