SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 7201

WEB KLIJENT ZA PREGLED I MANIPULACIJU IP ADRESA I DRUGIH MREŽNIH ARTEFAKATA

Karlo Lončar

Zagreb, svibanj 2021.

Sadržaj

Uvod 1

1. Analiziranje zapisa poslužitelja pogođenih DDoS napadima 2

1.1. DDoS napadi 2

1.2. Izrađivanje alata za analizu zapisa 4

2. Okvir za izradu korisničkog sučelja Angular 6

2.1. Povijest 6

2.2. Primjena 7

2.3. Struktura 9

2.4. Angular CLI 14

2.5. Angular Materials 15

3. Implementacija korisničkog sučelja 16

3.1. Izrada osnovne strukture aplikacije 20

3.2. Povezivanje komponenata preko usmjeravanja i modula 21

3.3. Komunikacija sa poslužiteljem 24

3.4. Formatiranje i vizualizacija podataka 28

3.5. Tumačenje vizualiziranih podataka 29

Zaključak 30

Literatura 31

Sažetak 32

Summary 33

Uvod

Otkako je internet postao najveći medij za komunikaciju između ljudi na svijetu, njegove mogućnosti i funkcije su se razvijale eksponencijalno. Stvari koje su se činile neizvedivim danas se puštaju u produkciju. Kao rezultat se počinje stvarati jedna mreža usluga i informacija koje čovjeku pružaju pomoć i alate za rješavanje modernih problema. Čovjeku se pruža mogućnost pregleda i upravljanje svojim podacima, financijama, putovanjem, edukacijom i drugim aspektima života koristeći napredne usluge koje mu pružaju potrebne mogućnosti gdje god se nalazio i kada god mu trebaju.

Isto kako čovjek uživa u koristnostima koje mu te usluge pružaju, tako postaje ovisan o njima. Kako ljudi ovise o web uslugama i servisima za svoje vlastite i poslovne potrebe, postavlja se pitanje kako će ti isti ljudi biti pogođeni kada im te usluge budu uskraćene? Internet je vrlo moćan alat, ali se uviđa da na isti način na koji se koristi za dobrobit čovječanstva, postoje ljudi koji iskorištavaju taj alat za svoje zlonamjerne ciljeve. Nakon mnogobrojnih zlonamjernih napada u zadnjih pedesetak godina, policija vodi konstantu borbu u suzbijanju napada, preventivnom zaštitom i istraživanjem napada kako bi saznali što više informacija o napadaču te ga priveli pravdi.

S tom namjerom se pojavila ideja da se olakša posao istražiteljima tako da bi se izradila aplikacija koja bi pomogla u vizualizaciji podataka nastalih od zapisa poslužitelja pogođenih napadom uskraćivanja usluga. Ideja je prikupiti te podatke i prikazati ih vizualno preko odnosa IP adresa i interakcija sa poslužiteljem na takav način da bi istražitelji mogli odmah prepoznati koje aktivnosti u napadu su bile od najvećeg značaja te bi mogli efikasnije i brže uputiti istragu u pravom smjeru.

Prvo poglavlje će objasniti kako se zapravo odvijaju napadi uskraćivanja usluga te će pobliže objasniti kako će ovaj alat izvući informacije iz tih situacija i kako će ih transformirati i konačno prikazati na prikladan način. U drugom poglavlju će se objasniti okvir za izradu tog alata Angular, kako je nastao, zašto je odabran za izradu ovakvog alata te njegova struktura i funkcionalnost. U trećem poglavlju će se objasniti proces izradnje ove aplikacije, kako postaviti dobre temelje aplikacije, povezati sve dijelove i funkcionalnosti te povezati sučelje sa poslužiteljem otkud će se primati obrađeni podatci te kako će biti vizualizirani u sučelju.

1. Analiziranje zapisa poslužitelja pogođenih DDoS napadima

Najčešći scenarij napada na poslužitelj se očituje kao neobično velika količina zahtjeva od velikog broja korisnika u kratkom vremenu. Postavlja se ogroman pritisak na infrastrukturu poslužitelja i njegovu mogućnost procesiranja zahtjeva. Kada poslužitelj nije u mogućnosti obraditi sve zahtjeve u dopuštenom vremenskom periodu govorimo da je njegova usluga uskraćena.

Nakon prikladnih postupaka sigurnosnih mjera pokušava se razriješiti i suzbiti napad usmjeren na poslužitelja. Nakon razrješenja takvog problema pokreće se istraga da se pokuša utvrditi razlog, motivacija i identitet napadača ili grupe napadača i najčešće sve što je istražiteljima na raspolaganju je zapis mrežnog prometa poslužitelja u vrijeme opterećenja spremljen u oblik dnevnika zapisa. Dnevnik zapisa sadrži informacije o IP adresama koje su stupile u komunikaciju sa poslužiteljom te što su zahtijevale od njega u vremenskom periodu opterećenja poslužitelja.

1.1. DDoS napadi

Napadi uskraćivanja usluga (engl. denial of service, DoS) opisuju sve aktivnosti koje dovode do preopterećenja mrežne infrastrukture poslužitelja u cilju njegovog privremenog ili trajnog onesposobljavanja [1]. DoS napad se karakterizira situacijom u kojoj napadač preko jedne IP adrese slijedno šalje zahtjeve obujmom i količinom većom od one za koju je poslužitelj namijenjen da podržava. Poslužitelj pokušava obraditi sve zahtjeve od napadača i troši većinu ili sve svoje resurse za to, a istovremeno nije više u mogućnosti pružiti zahtijevane resurse ostalim korisnicima zbog svoje preokupacije zahtjevima napadača te su normalni korisnici stavljeni u situaciju gdje predugo čekaju za obradu zahtjeva ili čak dobivaju poruke greške od poslužitelja kako je trenutno nedostupan. Na sreću ovakav oblik napada je primitivan i razvili su se obrambeni sustavi koji mogu raspoznati zlonamjernu IP adresu napadača te ju blokirati kako bi sačuvali resurse poslužitelja. Kako priča ne bi tu završila, postoje drugi sofisticiraniji napadi smišljeni od strane napadača na koje nije tako jednostavno reagirati i razriješiti.

Raspodijeljeni napadi uskraćivanja usluge (engl. distributed denial of service) su napadi u kojima napadač koristi više računala kako bi preopteretio poslužitelja. Kada pričamo o naprednim DDoS napadima treba uzeti u obzir da se često izvršava koordinirani napad od strane stotine tisuća računala odjednom. Pri pripremi napada, napadač pokušava zaraziti što više računala svojim virusom na razne načine od stavljanja malicioznih programa za preuzimanje na nesigurnim web stranicama, slanjem email poruka sa sumnjivim linkovima te psihološkom manipulacijom korisnika da posjete određene linkove i web stranice.

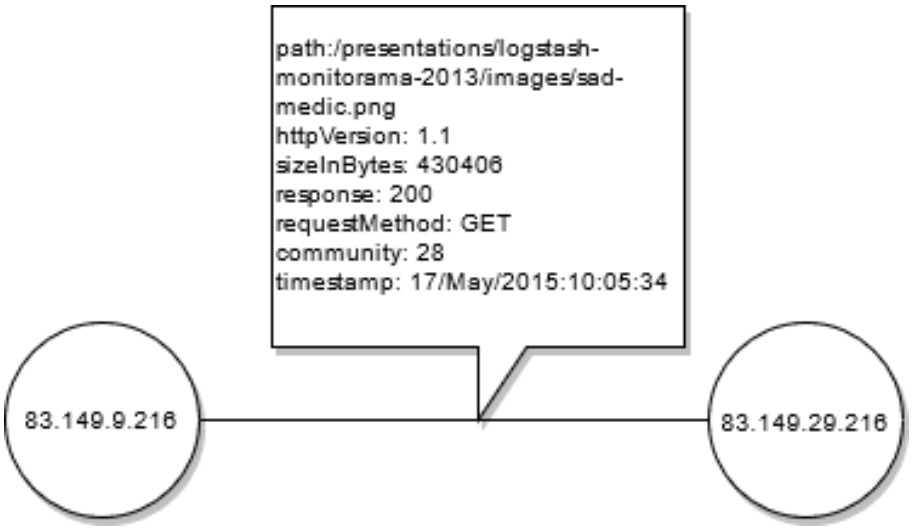
Ono što napadač u svojoj fazi pripreme pokušava je izgraditi mrežu zaraženih računala zvanu mreža kompromitiranih računala (engl. BotNet) te želi pridobiti potpunu kontrolu nad tim računalima. Kada je pridobio dovoljno računala na svoju stranu namješta automatiziran napad na određen poslužitelj, njegove dijelove ili na više poslužitelja odjednom. U trenutku pokretanja napada na poslužitelju se primjećuje neobična promjena u količini zahtjeva od mnogobrojnih jedinstvenih IP adresa. Takvi napadi su puno efektivniji i rade još veće opterećenje te stavljaju poslužitelj u stanje nepogodnosti gdje on ne može sa sigurnošću raspoznati koje su IP adrese zlonamjerne, a koje pripadaju normalnim korisnicima. DDoS napadi su nažalost vrlo štetni te često dovode poslužitelj u situaciju gdje se on mora privremeno ugasiti kako bi se suzbio napad. U tom slučaju napadač postiže ono što je i prvotno želio – uskraćivanje usluga poslužitelja normalnim korisnicima.

Jedan od najznačajnijih BotNet napada se dogodio 2016. godine zvan „Mirai“ [2]. Mirai BotNet je zahvatio preko 600,000 uređaja i onesposobio većinu internetskih usluga na istočnoj polovici SAD-a. Ono što se inicijalno predviđalo kao kibernetički napad na nacionalnoj razini od strane drugih država i njihovih tajnih organizacija je ustvari bio izveden od par studenata koji su testirali svoj virus u svrhu varanja u online igrici „Minecraft“. Zanimljivo je kako napad nije ciljao zaraziti računala već „Internet of things“ uređaje tj. uređaje koje se obično ne smatra pravim računalima, ali i dalje sadrže procesorske funkcije i pristup internetu kao što su kućni ruteri, sigurnosne kamere i slično. Ključno je da se takvi uređaji ne mogu zakrpati rješenjima (engl. patch) i popravcima na udaljenost te da se nalaze na fizički nedostupnim mjestima te ih takvim čini lakim za eksploataciju prvobitnog zaštitnog mehanizma koji je ugrađen u njih pri njihovoj izgradnji. Mirai DDoS napad je pokazao kako i najmanja eksploatacija najjednostavnijim postupcima može učiniti nezamislive štete.

Treba se uzeti u obzir da se ovakve situacije mogu pojaviti i bez loših namjera gdje se pristup web stranicama povećava prirodno, npr. na stranicama trgovina tijekom velikih popusta kao što su Crni Petak (engl. Black Friday). U razlikovanju prirodnih opterečenja od pravih napada uskraćivanja usluga bitna je nepredvidljivost događaja.

1.2. Izrađivanje alata za analizu zapisa

Razvitkom tehnologija otvaraju se novi uvidi i metodologije razrješavanja problema. Prvenstveno pri izradi alata za analizu podataka mora se shvatiti ovisnost i limitacija baze podataka na kojoj će se razvijati. Razvojem novog tipa baze podataka zvane „graf baza“ nastaje novi način prikaza podataka. Novonastala struktura graf baze sadrži čvorove i odnose prema čvorovima. Takav način prikaza se može iskoristiti kako bi se pokazale analizirane IP adrese te njihovi odnosi prema napadnutim poslužiteljima. Mogu se prikazati sve interakcije prema poslužitelju preko odnosa, uvidjeti koliko zahtjeva je slano te što su konkretno bili ti zahtjevi sadržajno. Prikaz interakcije IP adrese sa poslužiteljem prikazan je Sl. 1.1.



Sl. 1.1 Prikaz interakcije IP adrese sa poslužiteljem

Jedna od takvih baza je Neo4j koja je nastala 2010. godine [3]. Neo4j je graf baza podataka koja se temelji na teoriji grafova. Bazirana je na modelu podataka koje čine čvorovi, veze i svojstva. Glavne karakteristike baze su garantirana konzistentnost podataka, visoka razina dostupnosti podataka krajnjim korisnicima te jednostavno skaliranje; dodavanjem novih čvorova i veza nema znatnog usporenja sustava. Može se koristiti s različitim programskim jezicima kao što je Python što olakšava komunikaciju s odabranim poslužiteljem aplikacije koji se isto bazira na Pythonu.

Uz ovaj pristup može se izgraditi aplikacija s kojom bi korisnik mogao upravljati opisanim prikazom pregledavajući odnose IP adresa sa poslužiteljima te detaljnije po potrebi proučavati izabrane IP adrese. Pri izgradnji aplikacije ovakve tematike potrebno je izabrati pravo sučelje. S obzirom da se izrađuje jedna funkcionalnost analize zapisa, potrebno je uzeti okvir koji nudi brzu i jednostavnu izradu sučelja te istovremeno gledajući potencijal ove aplikacije mora omogućiti dobru potporu njenog proširenja. Okvir koji zadovoljava ove uvjete je Angular te je kao takav izabran za izgradnju sučelja aplikacije.

Osnovni dijelovi sučelja bi bili postavljanje (engl. upload) datoteka koje sadrže zapise poslužitelja pogođenih napadima, tablični pregled postavljenih datoteka i analiziranih IP adresa te vizualizacija IP adresa sa svojim odnosima čiji bi se prikaz mogao manipulirati u pravom vremenu.

2. Okvir za izradu korisničkog sučelja Angular

Angular je radni okvir (engl. framework) za izgradnju korisničkog sučelja web aplikacija koji ima zanimljivu povijest te veliku primjenu u poslovnom svijetu. Angular se karakterizira kao okvir koji ima strmu liniju učenja, ali veliku isplativost ulaganja vremena u njegovo savladavanje. Kao i ostali okviri, sadrži vlastitu strukturu i pravila izrade sučelja koji će biti pobliže objašnjeni u ostatku poglavlja.

2.1. Povijest

Prije svog službenog puštanja u produkciju, zaposlenik Google-a Miško Hevery je napravio začetak okvira kao „side project“ čiji je cilj bio olakšanje izgradnje web aplikacija za par svojih internih projekata [4]. 2010 godine se službeno pušta u produkciju kao okvir otvorenog koda (engl. open source) za korisničko sučelje pod nazivom AngularJS (Angular JavaScript). Bazirao se na arhitekturi MVC (engl. model view controller) te se sastoji od HTML (engl. hyper text markup language), CSS (cascading style sheets) i JavaScripta.

Kako AngularJS nije ispočetka bio zamišljen kao službeni projekt izrade sučelja, kroz par godina je naišao na probleme limitacije svoje osnovne strukture. Druga rješenja su počela pružati usluge za koje AngularJS nije bio namijenjen. Jedna od najvažnijih mogućnosti je bilo programiranje poprečne platforme (engl. cross platform) uz skalabilnost (engl. scalability) oblikovanja korisničkog sučelja za koji gosp. Hevery nikad nije ni planirao kada je započeo projekt. Kroz godine 2014.-2015. AngularJS doživljava svoje prvo veliko restrukturiranje.

Pojavom novih arhitektura kod drugih kompetitora, prvenstveno React.js arhitekture zvane MVVM (engl. model view viewmodel), zamišljena je nova struktura koja se bazira na komponentama. U restrukturiranju okvira nastaje AngularJS 2.0 koji se preimenuje u samo Angular i koristi se anotacija „major.minor.patch“ pri imenovanju verzija. Trenutna verzija je Angular 11 koji je korišten u ovom radu. Usputno se zamjenjuje JavaScript s TypeScriptom za pisanje logičkih funkcija sučelja.

TypeScript je proširenje JavaScripta, koji uzima iste funkcionalnosti, ali ih proširuje s novim konceptima kao što su tipovi (engl. type) i bolja mogućnost testiranja. Kako web preglednici ne mogu koristiti TypeScript, Angular automatski transformira TypeScript kod u JavaScript.

2.2. Primjena

Angular je u istraživanju 2020. godine postigao odlične rezultate [5]. Programeri ga ocjenjuju kao jednog od najboljih alata za izradu web stranica te korištenje TypeScripta kao trećeg najpopularnijeg programskog jezika za izradu web aplikacija. Najveća prednost TypeScripta je to što ima ugrađene provjere i testove koji se provode u pozadini dok se izrađuje kod. To omogućava brzo uočavanje i sprječavanje kasnijih problema već u ranim fazama razvoja aplikacija. Također uz pravilnu potporu web preglednika moguće je otklanjanje pogrešaka (engl. debug) TypeScript koda aplikacije izravno u web pregledniku što pomaže u boljem snalaženju, testiranju i izgradnji.

Prednost Angulara prema drugim okvirima je standardizirana struktura koda, brzo izgrađivanje aplikacija te mogućnost izrade većih projekata zbog dobre preraspodjele koda i njegove ponovne upotrebljivosti (engl. reusability). Činjenica da je Angular dio Google-ovog Long Term Support-a (LTS) označava Google-ov plan da nastavi podršku i izgradnju okvira u budućnosti, te namjeru da poveća njegovu upotrebu u svojim aplikacijama. Uz Angular nestaje potreba pisanja „getter“ i „setter“ funkcija jer je svaki objekt Plain Old JavaScript Object (POJO) koji omogućava manipulaciju objekata svim općenitim JavaScript funkcionalnostima.

Angular stavlja veliki fokus na konzistentnost pisanja koda jer razumije probleme i moguće troškove koji se dešavaju zbog nekonzistencije. Kod u Angularu je višekratno upotrebljiv zbog svoje podjele na komponente. Komponenta koja npr. izračunava statistiku na jednoj stranici može biti iskorištena za istu funkciju na drugoj stranici lakim uvozom (engl import). Uz konzistentnost i ponovnu upotrebljivost nastaje i poboljšanje u čitanju koda što uvelike povećava produktivnost programera.

S obzirom na prijašnje spomenute zanimljivosti, nije iznenađenje da se jedne od najpopularnijih web stranica i web aplikacija baziraju na Angularu. Stranice kao: Gmail, Paypal, Lego i UPS [6].

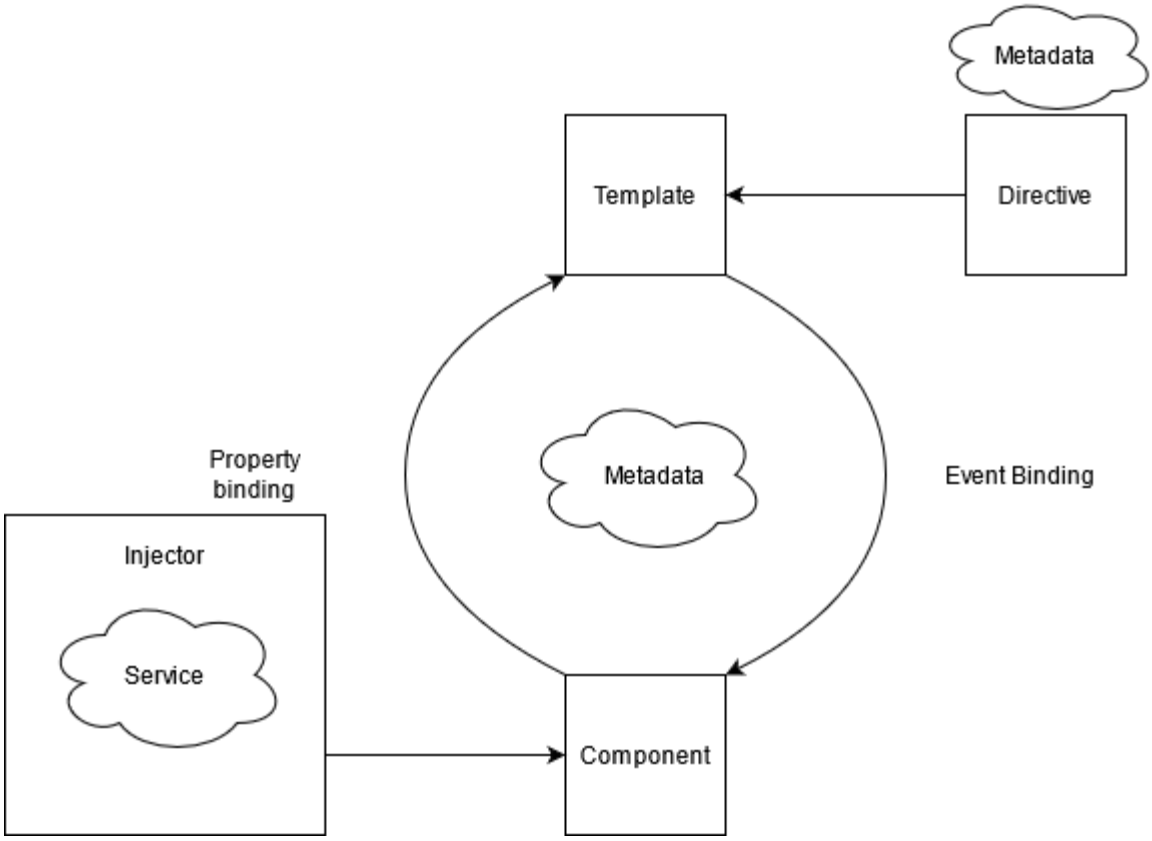
Angular pruža brzinu i odlične performanse u svojim aplikacijama. Pretvara predloške (engl. template) u visoko optimiziran kod za izvođenje preko JavaScript virtualnih strojeva što pruža koristnost pisanog koda te razinu produktivnosti radnog okvira. Također pruža potporu najkorištenijim poslužiteljskim jezicima kao što su „Node.js“, „.NET“ i „PHP“ za skoro instantno procesiranje HTML i CSS datoteka. Kako bi uštedio na resursima i poboljšao brzinu učitavanja stranica, Angular dijeli kod na dijelove koji se učitavaju po potrebi za procesiranje stranica koje se pregledavaju u tom trenutku, dok ostatak učitava u pozadini ili po potrebi.

Angular povećava produktivnost programera tako što im omogućava brzo slaganje elemenata prikaza uz jednostavnu i moćnu sintaksu predloška. Uz to se odlično integrira u većinu integriranih razvojnih okruženja (engl. integrated development environment, IDE) kako bi programeri dobili obavijesti o pogreškama tijekom pisanja koda, opcije automatiziranog ispunjavanja kodnih riječi i funkcija te ostale povratne informacije koje pomažu pri pisanju koda. Uz vlastito naredbeno korisničko sučelje (engl. command user interface, CLI) nudi ubrzane procese generiranja koda, posluživanja lokalnog poslužitelja, testiranja koda i slaganja koda aplikacije za produkciju.

2.3. Struktura

Arhitektura Angular aplikacije ovisi o određenim fundamentalnim konceptima koji će se objasniti u ovom poglavlju [7]. Osnovni gradivni blokovi Angular okvira su Angular komponente organizirane u „ngModules“. NgModule prikuplja srodni kod u funkcionalne skupine te definira aplikaciju kao skup povezanih ngModule-a. Aplikacija uvijek ima jedan korijenski modul (engl. root module) koji omogućuje prvotno pokretanje, te ostale module koji se nadograđuju na njega.

Kako je prije spomenuto, Angular uzima pristup izgradnje sučelja baziranog na komponentama. Komponenta je osnovna gradivna cjelina koja se pokušava izjednačiti s jednim prikazom na ekranu, funkcionalnom cjelinom ili cijelom stranicom. Komponente se umrežavaju gdje se bez poteškoća uspostavlja komunikacija i dijeljenje informacija između komponenata iste i različite upravljačke razine nastale granjanjem (engl. nesting). Komponente koriste servise koji pružaju specifične funkcionalnosti koje nisu direktno povezane s prikazom komponente. Prikaz pojednostavljene arhitekture prikazan je Sl. 2.1.



Sl. 2.1 Prikaz pojednostavljene arhitekture Angulara

Komunikacija podatkovne logike komponente i predloška koji se prikazuje na ekranu korisnika je pojednostavljena mehanizmima razmjene informacija, bazirana na MVVM arhitekturi. Ti mehanizmi su sljedeći:

- Event binding - je jednosmjerno povezivanje koje osluškuje promjene na predlošku i pri određenim promjenama omogućava pozivanje metoda i funkcija u komponenti te izmjenu vrijednosti varijabli u komponenti.
- Property binding - je jednosmjerno povezivanje gdje se podaci i vrijednosti varijabli prikazuju na elementima predloška. Interpolation je sličan property binding-u, ali je limitiran samo prikazom tekstualnih vrijednosti na predlošku.
- Two-way data binding - je dvosmjerno povezivanje koje sklapa property binding s event bindingom koristeći ngModel direktivu. Pri korisnikovom unosu podataka na predlošku dio komponente koji je vezan za taj dio automatski ažurira vrijednosti varijabli te ažurira elemente prikaza koji su vezani na te iste varijable komponente. Npr. korisnik unosi svoje ime i istovremeno se na ekranu ispisuje njegov unos imena.

Kako ne bi došlo do neurednosti koda u ovom procesu upravljanja podatcima, uvodi se pojam modula.

Modul (engl. module) obavlja funkciju upravitelja komponenata gdje oponaša funkcionalnu cjelinu grupacijom komponenti, servisa i direktiva te uvozom drugih modula. Cilj je sadržati funkcije na jednom skupu samo tamo gdje su potrebne. S tim pristupom se kod vrlo jednostavno organizira tijekom svog razvoja te se komunikacija komponenti pojednostavljuje podizanjem razine komunikacije na module. Komponente i dalje komuniciraju međusobno unutar jedne cjeline tj. modula, ali komunikacija s komponentama koje se nalaze na drugim stranicama ili nesrodnim funkcijama se odvija preko svojih modula.

Svaka Angular aplikacija sadrži barem jedan modul - korijenski modul, konvencionalno imenovan „AppModule“. Struktura modula je označena dekoratorom „ngModule“ koji označava datoteku kao modul. U dekoratoru možemo prilagoditi modul i povezivati ga s ostatkom aplikacije.

Osnovni opisni elementi „ngModule“ dekoratora su sljedeći:

- Declarations – tu se deklariraju komponente za koje će modul u pitanju biti zadužen upravljanjem.
- Exports – tu se upisuju imena komponenti za koje se želi da ih importaju drugi moduli i koriste u svojim funkcijama.
- Imports – tu se navode imena modula i komponenata koje se žele importati u modul kako bi ih on mogao koristiti.
- Providers – tu se definiraju servisi kako bi ih modul mogao koristiti unutar svojih deklariranih komponenti.
- Bootstrap – označava glavni prikaz aplikacije tj. korijenske komponente koja sadrži sve ostale prikaze i komponente. Samo korijenski modul bi trebao sadržavati „bootstrap“ svojstvo.

Servis (engl. service) je klasa koja je označena posebnim dekoratorom „@Injectable()“ te se koristi u slučajevima kada se želi upravljati podatcima ili programskom logikom koja nije nužno vezana za specifičnu komponentnu te se želi omogućiti njeno dijeljenje na više mjesta u aplikaciji po potrebi.

Pružatelji servisa mogu biti ubačeni (engl. injected) u komponentu kao ovisnost (engl. dependency) čime se izgrađuje modularan, ponovno uporaban i efikasan kod. Pri korištenju servisa koriste se određeni elementi koji upravljaju promjenama i događajima koji se događaju u aplikaciji. Angular je poznat po upotrebi specifične funkcionalnosti reagiranja na događaje. Dijelovi tih funkcionalnosti su oslušivaći, subjekti i mehanizam pretplate.

Oslušivaći (engl. observables) pružaju potporu u upravljanju porukama između dijelova aplikacije i izvan nje. Njihova primjena je učestala kao tehnika upravljanja događajima u Angularu, kod asinkronog programiranja i upravljanja s više vrijednosti istovremeno. Mehanizam oslušivaća je softverskog (engl. software) oblikovanja u kojem objekt, zvan subjekt (engl. subject), održava listu svojih ovisnih dijelova, zvanih oslušivaćima i automatski ih obavještava o nadziranim promjenama. Ovaj mehanizam je sličan „publish/subscribe“ dizajnerskom pristupu.

Ključan dio mehanizma je da promjene i informacije koje se prenose kao tok podataka (engl. data-stream), nisu uključene u proces izvođenja sve dok se ne pretplate (engl. subscribe) na taj oslušivać. Nakon pretplaćivanja preuzima se tok podataka i mogu se procesirati zamijećene promjene ili odgovori na zahtjeve.

Najčešća primjena je u sučeljima za programiranje aplikacija (engl. application programming interface, API) koje se koriste u razmjeni poruka između korisničkog sučelja i poslužitelja koji sačinjavaju aplikaciju gdje se zahtjevi sa sučelja šalju prema poslužitelju te se prate njihova stanja preko oslušivaća koji primaju odgovore od poslužitelja. Pretplativši se na te oslušivaće mogu se koristiti dohvaćene informacije s poslužitelja u funkcijama korisničkog sučelja.

Ubacivanje ovisnosti (engl. dependency injection) je pristup ugrađivanja servisa u komponente kako bi se odvojili zadatci koje izvršava sama komponenta i koje izvršavaju funkcije servisa. Prakticira se pristup da se svaka komunikacija sa poslužiteljem ili različitim drugim API-jem te internetskim uslugama provodi kroz ubačene servisne funkcije dok komponenta koristi samo rezultate tih servisnih funkcija kako bi upravljala svojim prikazima stranice.

Moduli, komponenete i servisi su zapravo klase koje koriste dekoratore (engl. decorators). Ovi dekoratori označavaju Angularovom sastavljaču (engl. compiler) o kakvom tipu datoteka se radi. Označavaju tip strukture te pružaju meta podatke (engl. metadata) koji označavaju kako će se oni koristiti i interaktirati u aplikaciji.

Meta podatci komuniciraju Angularu kako će procesirati klasu. Za primjer se uzima glavna komponenta *app.component.ts*:

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
```

U ovoj komponenti se može uočiti da `selector` označava Angular sastavljaču o kojoj komponenti se radi kada naiđe na istoimeni naziv u etiketi (engl. tag) HTML koda. Ono što Angular sastavljač u pozadini radi je da takvu specifičnu etiketu zamijeni HTML

datotekom označenoj u `templateUrl`. `styleUrls` definira CSS datoteku čije klase povezuje sa `templateUrl` datotekom radi izrade vlastito odabranog oblikovanja prikaza.

Jedna od najvećih prednosti Angulara su njegove direktive. Direktiva (engl. directive) je proširenje HTML etiketa s raznim prilagodljivim funkcionalnostima koje nude više opcija pri oblikovanju stranice. S obzirom na to da su prikazi u Angularu dinamični, pri procesiranju prikaza Angular prati instrukcije zadane direktivama. Direktive uvode mogućnost povezivanja korisničkog unosa s logikom programa i vrijednostima varijabli, prikazivanju i sakrivanju određenih HTML elemenata ovisno o vlastito zadanim uvjetima. Postoje dva tipa direktiva: strukturalna direktiva i atributna direktiva.

Strukturalna direktiva mijenja prikaz tako da dodaje, otklanja ili zamjenjuje određene elemente u prikazu. Karakterizira ju sintaksa: „*directiveName“ te su najčešći primjeri *ngIf i *ngFor.

Atributna direktiva mijenja izgled ili ponašanje određenog elementa. Izgleda kao obični HTML atribut te je tako i dobila ime. Primjer:

```
„<input [(ngModle)="user.name"]>
<p> This is your input:{{user.name}}</p>
```

`[(ngModel)]` direktiva uključuje dvostranu vezu podataka te u ovom primjeru osigurava da će se objekt `user` u komponenti mijenjati ovisno o onom što korisnik upiše. Time se osigurava automatsko ažuriranje unutarnjih varijabli korisničkim upisom te se ista takva varijabla može ispisivati na prikazu. Direktive uvode i opciju postavljanja cijelih komponenti unutar vlastitih etiketa bilo gdje u HTML datoteci. Npr. ako se izgradi kalkulator u jednoj komponenti s vlastitim predloškom, ta se komponenta može uvrstiti unutar *div* etikete te će se Angularov sastavljač pobrinuti da se taj kalkulator tamo pravilno smjesti sa svim svojim funkcionalnostima i oblikovanjem preko klasa CSS datoteke vezane za njega.

Za prikaz određenih elemenata i stranica tj. navigacijom aplikacije, brine se Angularov sistem usmjeravanja (engl. routing) koji definira kako će se i kada učitavati određene komponente. Jedno od velikih koristi kojeg omogućava usmjeritelj je koncept zvan „lazy-loading“. Umjesto da se sve datoteke i resursi učitavaju pri dolasku na aplikaciju, što bi usporilo njeno vrijeme rada, usmjeritelj ima opciju učitavanja samo onih modula i komponenti koje se trenutno prikazuju dok se ostatak aplikacije može učitavati u pozadini ili isto po potrebi. Tako se ubrzava rad aplikacije te nudi puno bolje korisničko iskustvo.

2.4. Angular CLI

Angular CLI je pomoćni alat razvijen od programera Angulara koji omogućava generiranje učestalog koda na brži i efikasniji način [8]. Nudi dodatne korisne opcije kao što je automatizirano uvođenje usmjeravanja komponenti, posluživanja privremenog lokalnog poslužitelja za pokretanje i izgradu aplikacije te mogućnost sastavljanja koda za produkciju. Preko Node.js biblioteke nudi široku opciju generiranja dijelova koda preko „ng“ funkcija. Npr. umjesto pisanja vlastitih HTML, CSS i TS dijelova komponente pod imenom „Component_A“, upisom naredbe „ng generate component Component_A“ automatski se generiraju te datoteke i spajaju s ostakom aplikacije.

Instalacija Angular CLI-a se izvršava u terminalu uz pomoć node.js-ovog menadžera paketa „npm“. Izvođenjem samo jedne komandne linije `„npm install -g @angular/cli“`, mogu se koristiti svi alati koje pruža Angular CLI.

U razvoju aplikacije najčešća komanda koja se koristi je `„ng serve“` s kojom se generira lokalni poslužitelj u pozadini koji će se koristiti za pokretanje sučelja te se svakom izmjenom u kodu automatski ažuriraju elementi stranice.

Kada je sučelje spremno za povezivanje sa poslužiteljem koristi se komanda `„ng build“` koja sklapa datoteke u zadani direktorij odakle će poslužitelj koristiti HTML i JavaScript datoteke za prikaz aplikacije.

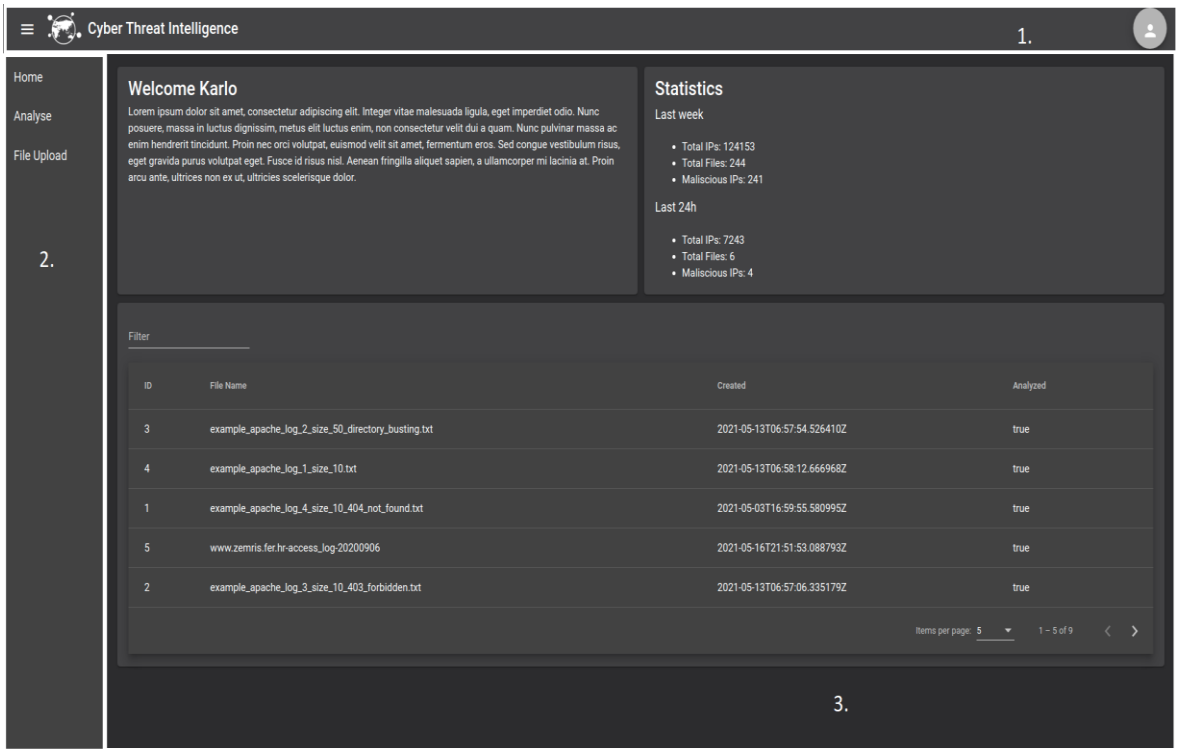
Zamjećuje se velika lakoća korištenja CLI-a koji obavlja složene zadatke spajanja datoteka, sklapanja koda te pokretanja aplikacije kroz par kratkih i jednostavnih linija koda.

2.5. Angular Materials

Angular Materials je biblioteka modula korisničkog sučelja izgrađena i održavana od strane Google-a koja uvodi vlastita rješenja u izradi oblikovanja sučelja aplikacije [9]. Nudi širok raspon alata za izradu tablica, gumbi, raspodjela elemenata stranice, prikaza kalendara i sličnih elemenata. Njihov je uvoz jednostavan te Google nudi odlične i jednostavne upute kako koristiti te module. Također nudi uvoz posebnih dijelova modula umjesto cijele biblioteke modula kako bi se iskoristili samo potrebni dijelovi. Uz dobro napisanu dokumentaciju s punu primjena za najčešće uporabe, svaki programer će s lakoćom moći izabrati, prilagoditi i implementirati materials elemente u svoje projekte. Angular Materials pruža jedinstveni izgled koji daje dojam modernog i poslovnog oblikovanja korisničkog sučelja.

3. Implementacija korisničkog sučelja

Pri izradi aplikacije dogovoren je raspored elemenata koji će biti jednostavan, ali će moći nuditi proširenja. Prateći princip „Single Page Application“ postavljen je material toolbar i material sidenav koji omogućava korisniku navigaciju po stranici. Toolbar i sidenav su elementi koji pripadaju glavnom AppModule-u koji će se prikazivati kroz cijelo sučelje. Prikaz material toolbara, sidebara i sadržaja prikazuje se Sl. 3.1.



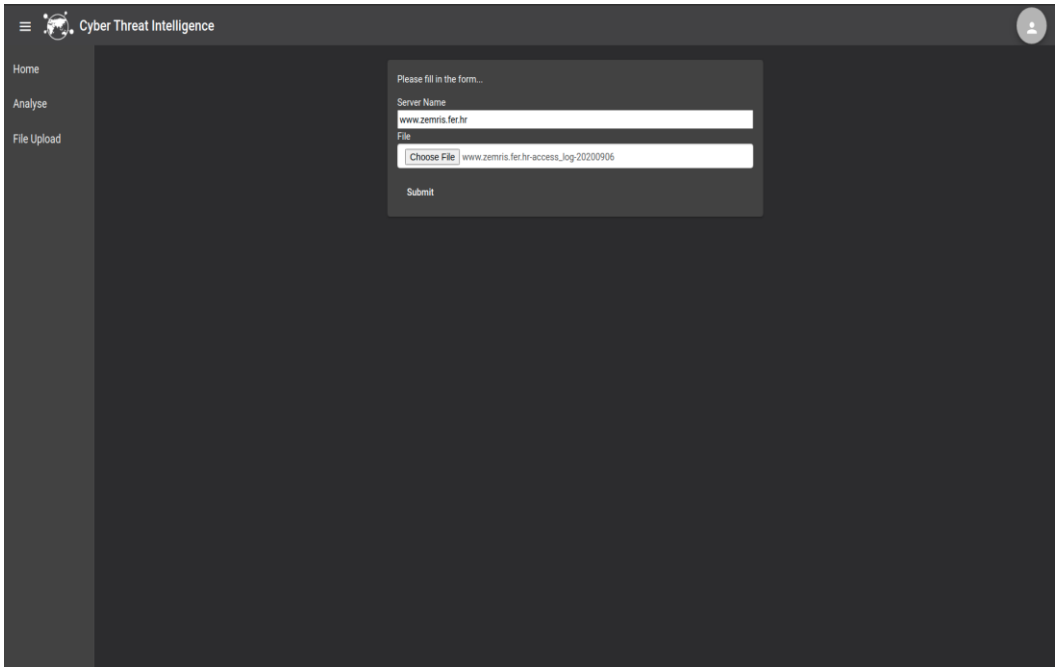
Sl. 3.1 Prikaz material toolbara, sidenava i sadržaja

Oznakom „1.“ označen je material toolbar koji sadrži gumb za proširenje sidenava sa odabirom predložaka te profilnom ikonom uz koju se korisnik može odjaviti iz aplikacije. Oznakom „2.“ označen je sidenavov sidebar u kojem je omogućena navigacija predložaka klikom na imena predložaka. Oznakom „3.“ označen je sadržaj sidenava gdje se učitavaju predlošci ovisno o navigaciji.

Iskorištava se funkcionalnost usmjeritelja koja će učitavati stranice po potrebi u sredini stranice. Korisnik dobiva dojam da aplikacija učitava različite stranice klikanjem ikona navigacije dok se zapravo samo sadržaj glavne komponente mijenja.

Kako bi iskoristili opciju lazy-loading-a treba se odlučiti koje će se komponente generirati i kako će se spajati [10].

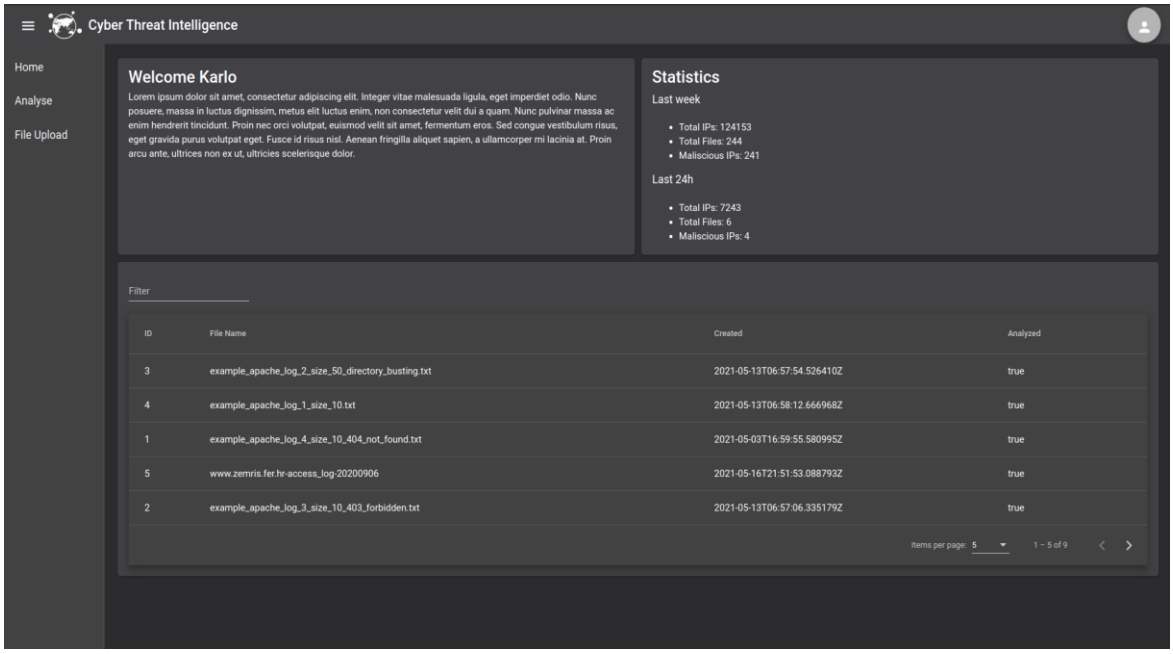
Za analizu IP adresa iz dnevnika zapisa zamišljena su tri pogleda . Prvi pogled je „File Upload“ pogled gdje će korisnik moći unijeti ime servera koji je napadnut DDoS napadom te učitati njegove zapise. Prikaz File Upload pogleda prikazan je Sl. 3.2.



Sl. 3.2 Prikaz File Upload pogleda

Za pravilno učitavanje datoteka koristio se koncept formulara preko modula „Reactive form“ [11]. Reactive form je modul koji omogućava izgradnju formulara, definiranja tipova podataka koji se očekuju od korisnika da će unijeti, te provjera (engl. validator) koje postavljaju uvjete prije prihvaćanja formulara. Koriste se ugrađeni validatori za provjeru imena servera, konkretno da ime nije kraće od 4 znakova i dulje od 24 znakova. Provjerava se unos datoteke, tj. korisnik mora učitati datoteku jer će se u suprotnom pojaviti poruke upozorenja i gumb za prihvat formulara će ostati onemogućen. Tek pravilnim unosom svih dijelova formulara i uspješnim provjerama će se omogućiti učitavanje datoteke na poslužitelj.

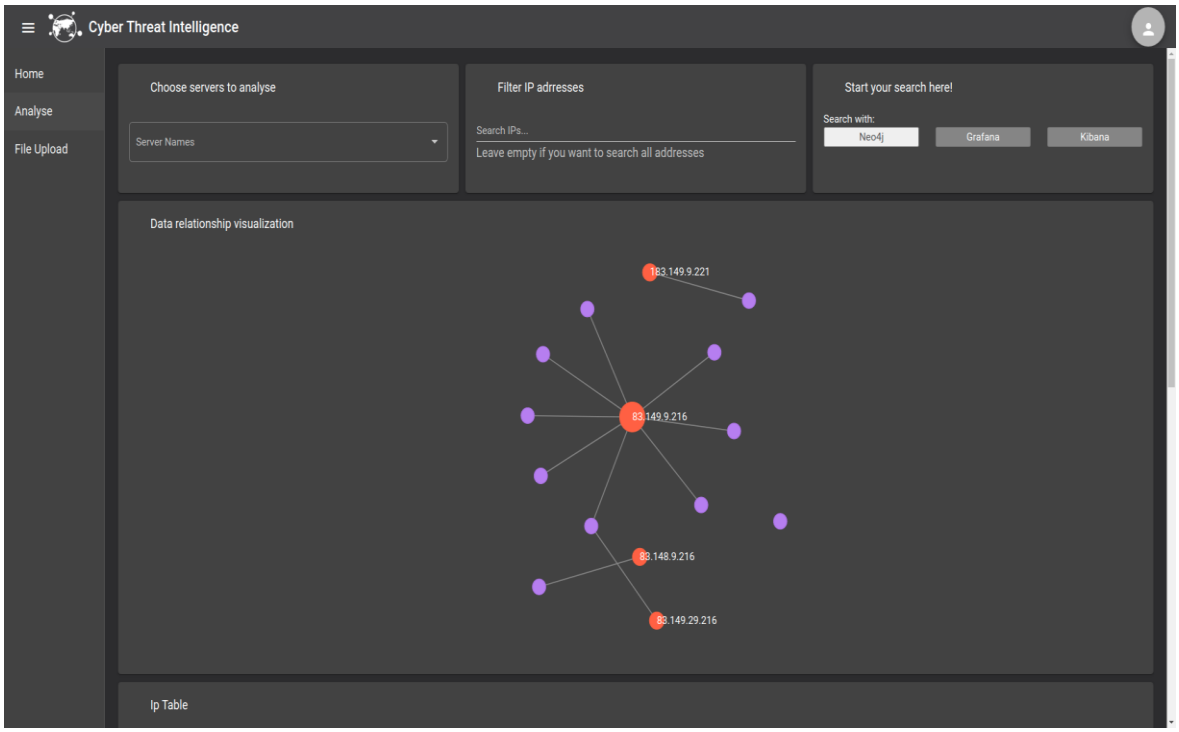
Drugi pogled je „Home“ pogled gdje korisnik može pregledavati sve dosad učitane datoteke i provjeriti jesu li analizirane od strane poslužitelja. Prikaz Home pogleda prikazan je Sl. 3.3.



Sl. 3.3 Prikaz Home pogleda

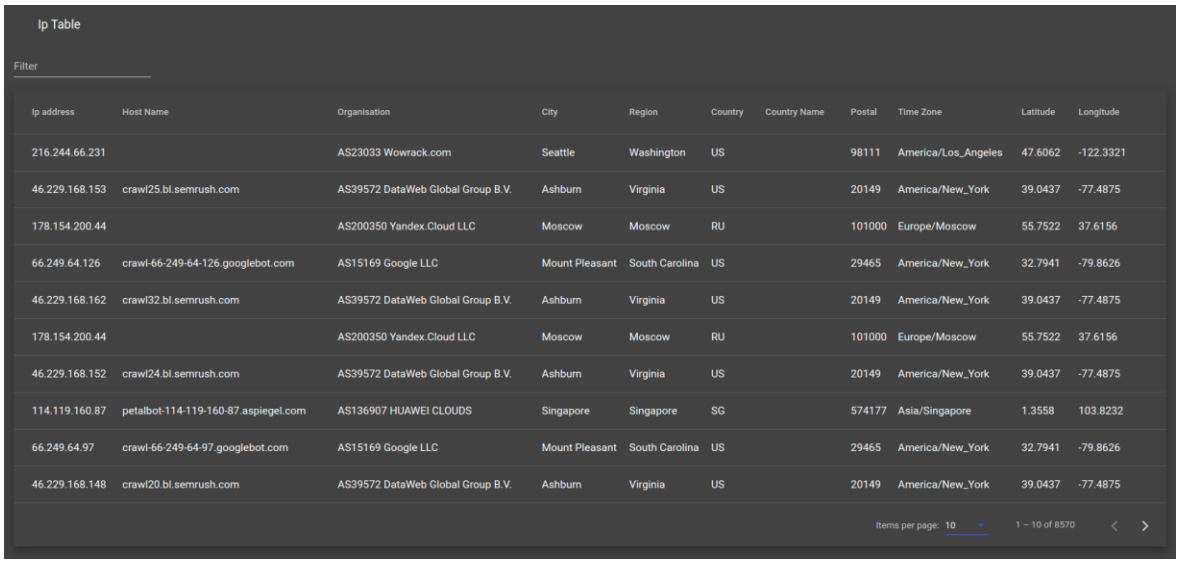
Na dnu prikaza se očituje tablični prikaz svih prijašnje učitanih datoteka sa svojim imenima, datumom učitavanja i statusom analize. Datoteka je analizirana ili nije što se očituje vrijednostima *true* ili *false*.

Treći i glavni pogled je „Analyse“ pogled u kojem će korisnik moći odabrati imena poslužitelja i IP adresa koje želi istraživati te će pretragom dohvatiti podatci sa poslužitelja. Vizualizacija pogleda se odvija preko skripte koju nudi d3.js servis. D3.js je JavaScript biblioteka za manipulaciju podataka. Koristi se za vizualizaciju velikih skupova podataka preko HTML-a, skalarnih grafičnih vektora (engl. scalable vector graphics, SVG) i CSS oblikovanja. Kao argumente prima informacije o čvorovima i njihovim vezama te ih prikazuje na sučelju. Prikaz Analyse pogleda prikazan je Sl. 3.4



Sl. 3.4 Prikaz Analyse pogleda

Klikom miša na izbornik u gornjem lijevom kutu na slici 3.4. korisniku se prikazuje lista imena servera koje je već postavio na poslužitelj. Nudi mu se i opcija filtriranja IP adresa i prije dohvaćanja informacija o njima sa servera te klikom miša na gumb pod nazivom „Neo4j“ u gornjem desnom kutu, pokreće se proces dohvata analiziranih podataka sa poslužitelja i njihova vizualizacija. Prikaz tablice analiziranih IP adresa prikazan je Sl. 3.5.



Sl. 3.5 Prikaz tablice analiziranih IP adresa

Tablica pruža bolji uvid u opće informacije analiziranih IP adresa. Korisnik ima mogućnost sortiranja svakog stupca klikom miša na ime stupca preko MatSort modula. Omogućeno je i filtriranje po bilo kojem dijelu tablice. Korisnik može filtrirati po IP adresama, imenima organizacija i slično.

3.1. Izrada osnovne strukture aplikacije

Pri izgradi aplikacije koristit će se mogućnosti Angular CLI-a. Kako bi se postavili temelji aplikacije u terminal se upisuje naredba „`ng generate new Ime_Aplikacije`". Ponude se opcije usmjeritelja i izbora oblikovanja gdje se odabire ugradnja osnovnog usmjerivanja i CSS način oblikovanja. Nakon kratkog vremena osnovni elementi su generirani od strane Angular CLI-a.

Pri sastavljanju koda sve datoteke se ugrađuju u *index.html* datoteku te *main.js* file. U *index.html* datoteci može se vidjeti da se u body etiketi nalazi etiketa koja sadrži ime aplikacije. To je glavna komponenta koja raspolaže ostatkom aplikacije i većina izmjena će se izvršavati na njoj. Sada se trebaju generirati komponente pogleda sa svojim modulima.

S komandom „`ng generate component home`" generira se home komponenta. Uvrštavajući imena „file-upload" i „analyse" umjesto „home" slično se stvaraju i ostale komponente. Preko CLI-a se generiraju njihovi moduli i usmjerenja naredbom „`ng generate module home –routing`", te kao i u generiranju komponenata, zamjenom imena generiraju se moduli i usmjerenja za preostale komponente.

U datoteci *app.component.html* slaže se osnovni prikaz stranice. Na vrhu stranice se dodaje prijašnje opisan material toolbar, te se na lijevoj strani dodaje material sidenav. Definira se mjesto gdje će se učitavati sve ostale stranice preko *router-outlet*-a i datoteke *app.component.html*:

```
<mat-sidenav-content class="sidenav-content">
      <div>
        <router-outlet></router-outlet>
      </div>
 </mat-sidenav-content>
```

Router-outlet dinamički uvrštava komponente ovisno o trenutnoj navigaciji sučelja koja će se uspostaviti u nastavku.

3.2. Povezivanje komponenata preko usmjeravanja i modula

U glavnom modulu usmjeravanja *app-routing.module.ts* mora se definirati korijen usmjeravanja na sljedeći način:

```
@NgModule({
      imports: [RouterModule.forRoot(routes)],
      exports: [RouterModule]
})
export class AppRoutingModule{ }
```

Funkcija `RouterModule.forRoot(routes)` govori aplikaciji da se sve ostale rute povezuju preko njega; to je početak svih ruta. Definiraju se sve rute jedinstvenih lokatora resursa (engl. uniform resource locator path) i poziva se metoda `loadChildren` koja omogućava lazy-loading. Npr. Definiranje rute za učitavanje stranice „analyse“:

```
Export const routes: Routes =
[
      {
      path: 'analyse',
      loadChildren: () => import('./analyse/analyse.module').
      then(m=>m.AnalyseModule)
      }
...
```

Na sličan način se definiraju rute za Home i File Upload stranice. Navigacijom na određeni dio sučelja glavni modul usmjeravanja prepušta modulu zaduženom za tu rutu da dalje navigira usmjeravanjem i da učitava komponente i dodatke vezane za nju. U specifičnom modulu postavljamo rutu za trenutnu komponentu i pružamo opciju ponovnog gniježđenja putem lazy-loading-a kako bi se u budućnosti mogla daljnja navigacija povezati novim dubljim rutama.

Npr. U datoteci *analyse-routing.module.ts* definiramo trenutnu rutu kao onu koja poslužuje AnalyseComponent:

```
Const routes: Routes = [
      {
      path: '',
      pathMatch:'full',
      component: AnalyseComponent
      }
];
@NgModule({
      imports: [RouterModule.forChild(routes)],
      exports: [RouterModule]
})
Export class AnalyseRoutingModule{ }
```

Omogućavanjem lazy-loading-a upravlja funkcija `RouterModule.forChild(routes)`. Također se mora napomenuti da se opcijom `pathMatch:'full'` definira čitanje ruta. Uz opciju *'full'* Angular sastavljač tumači da se prvo gleda ruta *analyse/* te se nakon toga gledaju rute definirane u *analyse-routing.module.ts* datoteci. Bez opcije *'full'* Angular sastavljač bi protumačio početnu korijensku rutu kao onu koja pripada analyse komponenti.

Nakraju se moduli usmjeravanja povezuju u module koji su zaduženi za njih. Npr. u datoteci *app.module.ts* uvrštava se AppRoutingModule:

```
@NgModule({
...
imports: [
      ...
      AppRoutingModule,
      ...
],
...})
Export class AppModule{ }
```

Nakon izrade osnovne strukture sučelja potrebno je uspostaviti komunikaciju sa poslužiteljem kako bi se učitavale datoteke i dohvaćali podatci, tj. kako bi aplikacija postala interaktivna. U nastavku će se objasniti spajanje sučelja sa puslužiteljem preko Angularovih ugrađenih alata.

3.3. Komunikacija sa poslužiteljem

Angular pojednostavljuje komunikaciju sa poslužiteljem preko svojih servisa. Potrebni servisi su postavljanje datoteka, dohvat svih prijašnje postavljenih datoteka te dohvat informacija o analiziranim IP adresama. Svi servisi su smješteni u *file.service.ts* te se definiraju na sljedeći način:

```
@Injectable({
  providedIn: 'root'
})
export class FileService {
  constructor(
    private http:HttpClient,
    private params:HttpParams
    ) { }
  uploadFile(formData : FormData, csrf:any){
     return this.http.post('/log/upload', formData, {
      headers: {
        'X-CSRFToken' : csrf
      }
    });
  }
  getServerNames(){
    return this.http.get("/servername", {responseType: 'json'});
  }
  getFilteredDataForMatTable(listOfServersQuery:string,
      searchIpAddressesQuery : string): Observable<IpAddress[]>{
    let params = new HttpParams();
    params = params.append('servernames', listOfServersQuery);
    params = params.append('ipaddresses', searchIpAddressesQuery);

    return this.http.get<IpAddress[]>('/ip',{
```

```
      responseType : 'json',
      params : params
    });
  }
  getFilteredDataForNeoNodes(listOfServersQuery:string,
searchIpAddressesQuery: string): Observable<any[]>{
    let params = new HttpParams();
    params = params.append('servernames', listOfServersQuery);
    params = params.append('ipaddresses', searchIpAddressesQuery);

    return this.http.get<any[]>('/d3/nodes',{
      responseType: 'json',
      params: params
    })
  }

  getFilteredDataForNeoAPIlinks(listOfServersQuery:string,
searchIpAddressesQuery: string): Observable<any[]>{
    let params = new HttpParams();
    params = params.append('servernames', listOfServersQuery);
    params = params.append('ipaddresses', searchIpAddressesQuery);

    return this.http.get<any[]>('/d3/links',{
      responseType: 'json',
      params: params
    })
  }
  getAllFiles():Observable<myFile[]>{
    return  this.http.get<myFile[]>("/logfiles", { responseType: "json"})
}}
```

Angular nudi specifične module koji se uvrštavaju kako bi se olakšalo upravljanje zahtjevima prema serveru. Koriste se moduli HttpClient, HttpParams i naknadno instaliran ngx-cookie-service uz komandnu liniju „`npm install -g ngx-cookie-service`“ koji će pomoći pri verifikaciji slanja zahtjeva. HttpClient modul se inicijalizira u konstruktoru i možemo ga koristiti u objektu u kojem je definiran. Podržava sve tipove zahtjeva kao što su GET, DELETE, PUT, POST i slično. HttpParams modul pruža lakšu modifikaciju headera, parametara i dodatnih opcija pri slaganju HTTP zahtjeva. Koristimo ngx-cookie-service kako bi dohvatili vlastiti kolačić (engl. cookie) generiran od strane poslužitelja te šaljemo isti za uspješnu verifikaciju.

Nakon definiranja servisa preko konstruktora (engl. constructor), servisi se uvrštavaju u komponente za uporabu. Kao primjer uzimamo servis dohvata informacija o IP adresama koje uvrštavamo u material tablicu te koristimo za vizualizaciju preko d3.js skipte u *analyse.component.ts* datoteci:

```
searchWithNeo4j(){
    const chosenServerNames = this.chosenServerNamesList.join(',');

    //Mat Table data request
    this.fileService.getFilteredDataForMatTable(chosenServerNames,
this.searchIpAddressesQuery)
      .subscribe((res)=>{
        this.dataSource = new MatTableDataSource(res);
        this.dataSource.paginator = this.paginator;
        this.dataSource.sort = this.sort;
      });

    //Neo4j nodes data request
    this.fileService.getFilteredDataForNeoNodes(chosenServerNames,
this.searchIpAddressesQuery)
      .subscribe((res)=>{
        this.nodes = res;
      });
```

```
    //Neo4j APIlinks data request
    this.fileService.getFilteredDataForNeoAPIlinks(chosenServerNames,
this.searchIpAddressesQuery)
      .subscribe((res)=>{
        this.APIlinks = res;
      });
    this.ngAfterContentInit();
  }
```

Preko poziva servisa te njegovom određenom metodom stvara se oslušivać rezultata. Nakon pretplaćivanja na oslušivać dobiva se rezultat u varijabli `res` koja sadrži sve informacije o IP adresama u obliku formata JavaScript objektne notacije (engl. JavaScript object notation, JSON) čija se vrijednost prosljeđuje u material tablicu kao tok podataka preko `this.dataSource = new MatTableDataSource(res);`. Varijabla `this.dataSource` se veže za prikaz preko prije spomenutog two-way binding-a kako bi se odmah ažurirao prikaz tablice na ekranu. U nastavku se pozivaju servisne funkcije za dohvat informacija za kreiranje čvorova i veza za vizualizaciju podataka preko d3.js skripte te se vrijednosti stavljaju u svoje namijenjene varijable, `this.nodes` za informacije o čvorovima i `this.APIlinks` za informacije o vezama između čvorova.

3.4. Formatiranje i vizualizacija podataka

Nakon dohvaćanja podataka sa poslužitelja preko prijašnjih servisnih funkcija te ubacivanja vrijednosti u namijenjene varijable pokreću se funkcije prikaza material tablice i d3.js skripte. Material tablica koristi princip dohvaćanja cijelog toka podataka te selektivno bira elemente koje će prikazati ovisno o svom material paginatoru tj. modulu koji je zadužen za broj prikaza elemenata po stranici tablice. Istovremeno novo dohvaćanje podataka pokreće novu instancu tablice koja koristi ista svojstva. Filtriranjem podataka prolazi se kroz sve elemente tablice te sastavlja novi prikaz sa samo odabranim elementima. Prednost ovakvog sistema je rijetkost komunikacije sa poslužiteljem jer nakon prvotnog dohvata podataka material tablica interno formulira svoj prikaz optimiziranim algoritmom i znatno brže se odvija njeno ažuriranje.

Uz Angularov proces detektiranja promjena pod nazivom „Angular Lifecycle-hook“ [12], d3.js gradi čvorove i veze za vizualizaciju. Internim provjerama promjena, preko lifecycle-hook-a, skripta kontinuirano iscrtava grafički prikaz. Korisnikovom interakcijom s elementima grafa stvara se dojam interakcije te se pružaju dodatne informacije postavljanjem miša na pojedini čvor ili vezu između dva čvora za detaljniji prikaz informacija.

3.5. Tumačenje vizualiziranih podataka

Nakon izgradnje aplikacije korisnik može lakše i jasnije istraživati grafove s IP adresama i njihovim odnosima. Korisnik bi ubrzo mogao razjasniti koje IP adrese imaju značajniju aktivnost od drugih. Pojedine IP adrese će se moći razlučiti po broju zahtjeva prema poslužitelju, po količini interakcija prema više poslužitelja pogođenih napadom uskraćivanja usluga te konačno pregledom sadržaja svakog zahtjeva. Također je bitno vrijeme svakog zahtjeva koji je upućen serveru.

Npr., intuitivno je zaključiti da čovjek nije sposoban poslati 100 zahtjeva za navigacijom na pojedine dijelove stranice servera ispod 1 sekunde. Takvi zahtjevi se mogu protumačiti kao automatizirani proces zlonamjernog programa da preoptereti poslužitelj s velikom količinom zahtjeva te bi takve IP adrese trebale biti stavljene u fokus kriminalne istrage.

Zaključak

Napadi uskraćivanja usluga predstavljaju veliki problem u sigurnosti. Posebno se posvećuju vrijeme i resursi za izradu boljih obrambenih sustava i poslužitelja s boljom mrežnom infrastrukturom, ali to je samo liječenje simptoma umjesto cijele bolesti. Veći fokus bi se trebao staviti na istraživanje razloga napada i pronalaska identiteta napadača analiziranjem IP adresa koje su imale kontakt sa poslužiteljem tijekom perioda napada. U ovom radu se izradio alat koji omogućava istražiteljima bolji pregled zapisa poslužitelja nakon napada. Aplikacija pruža bolji pregled cijele situacije kako bi istražitelji mogli lakše usmjeriti svoju istragu u pravom smjeru. Za izradu korisničkog sučelja korišten je radni okvir Angular koji se dokazao kao odličan izbor za izgradnju jednostavne aplikacije u kratkom vremenu. Sučelje pruža mogućnost postavljanja datoteka koje sadrže zapise interakcija IP adresa sa poslužiteljem te nakon analize tih zapisa na poslužitelju aplikacije, sučelje dohvaća te podatke i nudi pregledniji prikaz tih informacija tabličnim prikazima te izgradnjom vizualnog prikaza grafa. U grafu čvorovi predstavljaju pojedine IP adrese, a međusobne veze označavaju opis zahtjeva između IP adresa.

Aplikacija bi još mogla biti proširena drugim alatima za analizu podataka koji bi se jednostavno ugradili kao novi predlošci. Temeljna struktura sučelja dopušta proširenje novim komponentama koje se lako mogu povezati s ostatkom aplikacije. Predlaže se dodavanje detaljnijeg prikaza pojedinih IP adresa klikom miša na njihov identifikator u tablici ili u čvoru vizualiziranog grafa. Uz to bi bilo odlično izraditi ispis tog detaljnijeg prikaza u datoteku pdf formata.

Literatura

[1] Tehnical White Paper – Guide to DDoS Attacks, https://www.cisecurity.org/white-papers/technical-white-paper-guide-to-ddos-attacks/, pristup: 13.5.2021.

[2] Fruhlinger J.,The Mirai botnet explained, https://www.csoonline.com/article/3258748/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html, pristup: 13.5.2021.

[3] Sekhon S., What is Neo4j?,https://dev.to/sukhbirsekhon/what-is-neo4j-8jc, pristup: 13.5.2021.

[4] Gavigan D., The History of Angular, https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7, pristup: 13.5.2021.

[5] 2020 Developer Survey, https://insights.stackoverflow.com/survey/2020#technology-web-frameworks-all-respondents2, pristup: 13.5.2021.

[6] 10 Mostu Popular Angular Websites of 2020, https://wiredelta.com/10-most-popular-angular-websites-of-2020/, pristup: 13.5.2021.

[7] Murray, Nate, et al. "ng-book." The complete guide to Angular. Fullstack. io (2018), pristup: 13.5.2021.

[8] CLI Overview and Command Reference, https://angular.io/cli, pristup: 13.5.2021.

[9] Angular Material – Material Design components for Angular, https://material.angular.io/, pristup: 13.5.2021.

[10] Reis T., How to implement Lazy Loading in Angular, https://medium.com/@thiago.reis/how-to-implement-lazy-loading-in-angular-c8dcbf165561, pristup: 13.5.2021.

[11] Reactive forms, https://angular.io/guide/reactive-forms, pristup: 13.5.2021.

[12] Lifecycle hooks, https://angular.io/guide/lifecycle-hooks, pristup: 13.5.2021.

Sažetak

Web klijent za pregled i manipulaciju ip adresa i drugih mrežnih artefakata

U ovom radu je obrađena tema izrade korisničkog sučelja za analiziranje zapisa poslužitelja pogođenih napadima uskraćivanja usluga. Na početku su objašnjeni napadi uskraćivanja usluga te važnost izrade alata koji bi pomogao u istraživanju takvih napada. Zatim je obrazložen odabir radnog okvira Angular za izradu korisničkog sučelja aplikacije. Opisana je povijest i primjena radnog okvira te njegova struktura. U nastavku je napisan detaljniji proces izgradnje sučelja te prikazan način pisanja koda pri oblikovanju izgleda i izgradnji funkcionalnosti radnog okvira. Na kraju je objašnjen zamišljeni način korištenja sučelja te kako je aplikacija povezana u cjelinu.

Ključne riječi: obavještajni rad, kibernetički napadi, IP adresa, Angular, AngularJS, Graf baza Neo4j, Angular Materials

Summary

Web client for review and manipulation of IP addresses and other network artefacts

This paper deals with the topic of creating a user interface for analyzing server records affected by denial of service attacks. At the beginning, denial of services attacks are given an explanation, as well as the importance of developing tools to help investigate these attacks. The choice of the Angular framework for creating the application user interface is then explained. The history and application of the framework is described as well as it's structure. Furthermore, a detailed description is given of the process of building the interface and insight is given in how to write code when designing the layout and building the frameworks functionality. Finally, the designed way of using the interface is explained as the application is connected throughout.

Keywords: intelligence, cyber attacks, IP address, Angular, AngularJS, Graph database Neo4j, Angular Materials