SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3157

Napadi uskraćivanja usluge na PSD2 API i zaštita bez prekidanja TLS tunela

Marko Zaninović

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 3157

Napadi uskraćivanja usluge na PSD2 API i zaštita bez prekidanja TLS tunela

Marko Zaninović

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

DIPLOMSKI ZADATAK br. 3157

Pristupnik: **Marko Zaninović (0036509958)**

Studij: Računarstvo

Profil: Računarska znanost

Mentor: izv. prof. dr. sc. Stjepan Groš

Zadatak: **Napadi uskraćivanja usluge na PSD2 API i zaštita bez prekidanja TLS tunela**

Opis zadatka:

PSD2 je direktiva Europske unije koja je stupila na snagu u rujnu 2019. godine. Navedenom direktivom Europska unija traži od financijskih institucija, prvenstveno banaka, da omoguće pristup svojim poslovnim sustavima putem API-ja kako bi se omogućilo nuđenje financijskih usluga svim tvrtkama te na taj način liberaliziralo financijsko tržište. Međutim, otvaranjem novog komunikacijskog kanala koji zadire u samu jezgru IT sustava financijskih institucija otvara novi potencijalni vektor napada o kojemu treba voditi računa. U sklopu diplomskog rada potrebno je definirati scenarije napada uskraćivanja usluge koji se mogu izvršiti na izložen API. Potrebno je definirati moguće zaštite s posebnim naglaskom na zaštitu u oblaku. S obzirom na osjetljivost podataka koji se prenose preko PSD2 API-ja potrebno je istražiti mogućnosti pružanja zaštite bez prekidanja TLS tunela. Eksperimentalno verificirati predložena rješenja. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 23. lipnja 2023.

Zahvaljujem svom mentoru, izv. prof. dr. sc. Stjepanu Grošu, na pomoći u izradi ovog diplomskog rada.

Sadržaj

1. Uvod

Preko PSD2 direktive (eng. *Payment Services Directive 2*) Europska unija traži od financijskih institucija, prvenstveno ovdje od banaka, da putem svojeg API-ja omoguće pristup svojim poslovnim sustavima kako bi se svim tvrtkama omogućilo nuđenje financijskih usluga. Od kada je ta direktiva stupila na snagu u rujnu 2019. godine, ona je otvorila vrata uslugama i inovacijama koje mogu učiniti bankovne transakcije u Europskoj uniji isplativijim, lakšim i sigurnijim.

Međutim, stvaranjem tog novog komunikacijskog kanala financijske institucije se otvaraju novom potencijalnom vektoru napada koji ne smije ostati ignoriran. Cilj ovog rada je napraviti pregled napada uskraćivanja usluga (*eng. DoS, Denial of Service*) i istražiti kako se od njih zaštititi. Cilj tih napada je uskraćivanje usluga legitimnim korisnicima iscrpljivajući dostupne resursa žrtve slanjem velike količine internetskog prometa.

S obzirom da je tu riječ o osjetljivim podacima, kada bi treća strana ponudila zaštitu od napada tim financijskim institucijama pretpostavlja se da bi bilo poželjno da to rješenje nema pristup osjetljivim podacima u čitljivom obliku. Zbog toga će se u ovom radu razmotriti rješenja koja mogu štititi sustav bez prekida TLS tunela kojim se šalju osjetljivi kriptirani podaci.

Ovaj rad je organiziran na sljedeća poglavlja. U drugom poglavlju je napravljen pregled napada uskraćivanja usluga i detaljniji opisi napada od kojih bi se moglo obraniti. Jedno od mogućih rješenja koje ne prekida TLS tunel je korištenje obrnutog posrednika (*eng. reverse proxy*) koji je detaljnije objašnjen u trećem poglavlju. Četvrto poglavlje bavi se testiranjem posrednika koristeći klijentsku skriptu koja šalje zahtjeve koristeći Sandbox API Erste Banke i skripte pomoću koje se mogu izvršavati različiti napadi. U petom poglavlju se nalazi zaključak diplomskog rada i u šestom poglavlju se nalazi popis literature korištene u ovom radu.

2. Pregled napada uskraćivanja usluga

Napad uskraćivanja usluge je zlonamjeran pokušaj prekida normalnog prometa ciljanog poslužitelja, usluge ili mreže preplavljivanjem cilja ili njegove okolne infrastrukture internetskim prometom. Raspodijeljeni napadi uskraćivanja usluga (*eng. Distributed Denial of Service attacks, DDoS*) se provode pomoću mreže strojeva povezanih s internetom. Te se mreže sastoje od računala i drugih uređaja koji su zaraženi zlonamjernim softverom, što omogućuje napadaču da ih koristi u svrhu napada i zato se ovi napadi nazivaju raspodijeljenim napadima. Ovi pojedinačni uređaji se nazivaju botovi (ili zombiji), a grupa botova naziva se *botnet* [1].

Napad uskraćivanja usluge može imati sljedeći tijek:

Napadač kompromitira mrežu računala ili uređaja tako da ih zarazi zlonamjernim softverom ili dobije neovlašteni pristup. Ovi kompromitirani uređaji postaju dio botneta, kojim upravlja napadač. Napadač koristi zapovjedno-kontrolnu infrastrukturu (*eng. Comand and control*) za komunikaciju s ugroženim uređajima u botnetu. To napadaču omogućuje koordinaciju napada i izdavanje uputa članovima botneta. Napadač zatim daje upute kompromitiranim uređajima u botnetu da pošalju ogromnu količinu prometa ciljnom sustavu ili mreži. Takav generirani dolazni promet s kompromitiranih uređaja zasićuje mrežnu infrastrukturu mete trošeći dostupne resurse poput propusnosti, procesorske snage ili memorije. Kao rezultat toga, ciljni sustav ili mreža postaju nesposobni rukovati legitimnim prometom, uzrokujući degradaciju usluge ili potpuni prekid usluge. Napadi uskraćivanja usluga se mogu kategorizirati na napade na razini transportnog i napade na razini aplikacijskog sloja OSI modela [2].

2.1. Napadi uskraćivanja usluga na aplikacijskom sloju

Napadi aplikacijskog sloja koriste složenije mehanizme za napad. Umjesto da preplave mrežu prometom ili sjednicama, ovi tipovi napada ciljaju određene aplikacije i usluge kako bi polako iscrpili resurse na aplikacijskom sloju OSI modela. Napadi uskraćivanja na aplikacijskom sloju općenito ciljaju na HTTP protokol s svrhom iscrpljivanja ograničenih resursa dostupnih web servisima pa tako i kod REST API-ja banaka jer REST obično koristi HTTP kao svoj temeljni protokol koji donosi uobičajeni skup sigurnosnih problema.

Napadi na aplikacijskom sloju mogu biti vrlo učinkoviti pri niskim brzinama prometa, a promet uključen u napade može biti legitiman iz perspektive protokola. Zbog toga je teže otkriti napade na aplikacijskom sloju nego kod drugih vrsta napada uskraćivanja usluga, a bez prekidanja enkripcije njihovo otkrivanje nije moguće pa dolje navedeni napadi iz ovog potpoglavlja nisu bili testirani u sklopu ovog rada.

2.2.1. HTTP flood napad

HTTP flood napad je vrsta napada koja cilja web poslužitelje i cilj ovog napada je iscrpiti resurse poslužitelja kao što su to procesor, memorija i propusnost mreže čineći ga nesposobnim

odgovoriti na legitimne korisničke zahtjeve. U tipičnom HTTP flood napadu napadač koristi mrežu kompromitiranih računala kako bi generirao što veći broj HTTP zahtjeva prema ciljanom web poslužitelju. Ti su zahtjevi obično izrađeni tako da izgledaju kao legitimni promet što otežava poslužitelju razlikovanje pravih korisničkih zahtjeva od zlonamjernog skupa zahtjeva. Napadač šalje veliki broj HTTP GET ili POST zahtjeva često ciljajući određene URL-ove ili krajnje točke aplikacije koje isto može s vremenom mijenjati kako bi izbjegao detekciju. Zatrpavanjem poslužitelja brojnim zahtjevima on postaje nesposoban obraditi ih sve što rezultira degradacijom usluge ili njenom potpunom nedostupnošću.

HTTP flood napadi mogu biti vrlo štetni jer mogu poremetiti online usluge, utjecati na korisničko iskustvo i dovesti do financijskih gubitaka za tvrtke. Vlasnici i administratori web stranica mogu primijeniti različite protumjere za ublažavanje utjecaja HTTP flood napada kao što je ograničavanje brzine, filtriranje prometa i korištenje mreža za isporuku sadržaja [3] (*eng. CDN, Content Delivery Network*) za distribuciju prometa.

2.2.2. Napad korištenjem višestrukih HTTP glagola/metoda

Metode HTTP zahtjeva pokreću radnju na strani poslužitelja i zbog toga se mogu zvati HTTP glagolima. Uobičajeno je da jedan zahtjev sadrži jednu radnju ili glagol ali HTTP ima nešto manje poznatu opciju koja omogućuje korisnicima dodavanje više metoda u jedan HTTP zahtjev. Koristeći ovo znanje napadači žele što više posla prebaciti na poslužitelja u odnosu na broj zahtjeva koje moraju poslati. To znači da napadači mogu staviti više metoda u jedan zahtjev i poslati ga poslužitelju koji je prisiljen izvršiti sve metode koji se traže, što nije slučaj kod uobičajenih zahtjeva. Prednost koju ovo predstavlja napadačima je da se opseg napada može smanjiti u velikoj mjeri, što im pomaže da zbog tog manjeg prometa izbjegnu njihovo otkrivanje.

2.2.3. Slowloris napad

Slowloris je napad uskraćivanja usluga na aplikacijskom sloju koji djeluje korištenjem djelomičnih HTTP zahtjeva. Napad funkcionira tako da otvara veze s ciljanim poslužiteljem, a zatim te veze drži otvorenima koliko god može. HTTP je dizajniran imajući na umu korisnike s malom propusnošću pa zbog toga HTTP svojim korisnicima omogućuje fragmentaciju HTTP poruke u više paketa. Napadač koji fragmentira svoje HTTP poruke u iznimno male pakete može držati vezu otvorenom proizvoljno dugo vremena. Budući da web aplikacije imaju unaprijed definiran ograničeni broj pristupnih točki koje mogu održavati istovremeno, napadač koji uspije držati više veza otvorenima beskonačno dugo može prisiliti poslužitelja da više ne može uspostavljati nove veze s legitimnim korisnicima.

Koraci ovog napada su sljedeći:

- Napadač otvara što je veći mogući broj veza s poslužiteljem slanjem HTTP zahtjeva s djelomičnim zaglavljem i za svaki dolazni zahtjev se otvara nova dretva koja će se zatvoriti nakon što je veza dovršena.

- Kako ne bi došlo do isteka vremenskog ograničenja veze, napadač povremeno šalje nova djelomična zaglavlja tog zahtjeva poslužitelju kako on ne bi prekinuo vezu i održao ju živom.
- Ciljani poslužitelj ne može osloboditi te veze od kojih čeka kraj zahtjeva i kada su zauzete sve moguće pristupne točke dolazi do uskraćivanjem usluge jer poslužitelj više neće moći odgovarati na zahtjeve.

Ispis 2.1 prikazuje kako izgleda jedan legitimni zahtjev [4]. HTTP koristi povratni redak (*eng. CRLF, Carriage Return Line Feed*) za označavanje sljedećeg retka i koristi dva CRLF znaka za označavanje praznog retka. Obično prazan redak označava kraj zaglavlja u HTTP zahtjevu. Napadač može jednostavno izostaviti dvostruku CRLF sekvencu tako da poslužitelj pretpostavi da korisnik nije završio sa slanjem zaglavlja.

```
GET /index.php HTTP/1.1 [ CRLF]
Pragma : no-cache [CRLF]
Cache-Control : no-cache [CRLF]
Host : testphp.vulnweb.com[CRLF]
Connection : Keep-alive [CRLF]
Accept-Encoding : gzip, deflate [CRLF]
User-Agent : Mozilla / 5.0
(Windows NT 6.1 ; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko )
Chrome/28.0.1500.63 Safari / 537.36 [CRLF]
Accept : * / * [CRLF] [CRLF]
```

Ispis 2.1. Primjer legitimnog HTTP zahtjeva

2.2.4. Napad ubacivanja SQL izraza

Velik broj web-aplikacija radi s nekom verzijom SQL jezika (*eng. Structured Query Language*) za obradu podataka. Za ove aplikacije ubacivanje SQL izraza je ozbiljna prijetnja. Ubacivanje SQL-a obično je povezano s ubacivanjem zlonamjernih podataka ili curenjem osjetljivih informacija, ali također se može koristiti za uskraćivanje usluge jer napadač može potencijalno natjerati poslužitelja da izvrši radnje koje dovode do njegovog nekonzistentnog stanja ili čak da obriše cijelu bazu podataka bez koje sustav onda više ne može funkcionirati što onda dovodi do uskraćivanja usluge.

Do uskraćivanja usluge bi moglo doći na sljedeći način [4]. U primjeru SQL upita koji vrši poslužitelj nad svojom bazom podataka (Ispis 2.2) unos predstavlja vrijednost koje daje korisnik i ako pod unos predamo SQL izraz `'1;DROP TABLE zaposlenici'` SQL upit bi izgledao kako je to prikazano na ispisu 2.3. Zbog znaka ';' u našem ulazu koji označava kraj upita izvršiti će se upit u kojemu se bira zaposlenik ovisno o „ZapBr“ i dodatno će se izvršiti upit gdje će se izbrisati svi zapisi unutar tablice „zaposlenici“.

```
SELECT * FROM zaposlenici
WHERE ZapBr = $unos$;
```

Ispis 2.2 Primjer SQL upita

```
SELECT * FROM zaposlenici
WHERE ZapBr = 1;
DROP TABLE zaposlenici;
```

Ispis 2.3 Primjer konačnog SQL upita sa ubačenim SQL izrazom

Sprječavanje napada ubacivanja SQL-a zahtjeva korištenje sigurnih praksi pri implementaciji i usvajanje obrambenih mjera na različitim razinama aplikacije. Za sprječavanje ovakvih napade postoje sljedeće mjere i postupci:

- Korištenje parametriziranih upita ili pripremljenih izjava s rezerviranim mjestima umjesto dinamičke konstrukcije SQL upita. Ovaj pristup osigurava da se korisnički unos tretira kao podatak, a ne kao izvršni kod, čime se učinkovito sprječava SQL ubacivanje.
- Provođenje stroge provjere valjanosti podataka koji se unose kako bi se osiguralo da se unos pridrža očekivanog formata i tipa podataka. Iz unosa bi se trebalo ukloniti ili izbjegavati posebni znakovi koji se mogu koristiti za izmjenu SQL upita.
- Primjena načela najmanje privilegije na račune baze podataka i ograničavanje njihovih dozvola samo na ono što je neophodno za njihov rad. Isto se preporuča izbjegavanje korištenja povlaštenih računa baze podataka u kodu aplikacije.
- Korištenje pohranjene procedure umjesto ugrađivanja SQL upita izravno u kod aplikacije. Ovo pruža dodatni sloj sigurnosti osiguravajući da je logika baze podataka odvojena od koda aplikacije.

2.2. Napadi uskraćivanja usluga na transportnom sloju

S obzirom da TLS djeluje iznad transportnog sloja OSI modela sve do njega uključujući transportni sloj nije šifrirano. Napadi na ovoj razini uključuju preplavljivanja različitim TCP/UDP paketima. Ovdje ne trebamo imati na uvid osjetljive podatke korisnika već detektiramo napade promatrajući protokole koji se koriste i njihove atribute kao što su to zastavice koje se koriste u komunikaciji ili pomoću nekih drugih hardverskih rješenja.

2.2.1. SYN flood napad

SYN flood je o vrsti napada uskraćivanja usluge čija je svrha učiniti poslužitelj nedostupnim legitimnom prometu konzumiranjem svih dostupnih veza poslužitelja. SYN flood napad funkcionira iskorištavajući proces sinkronizacije u tri koraka protokola TCP.

Koraci u uspostavi TCP veza se uspostavlja su sljedeći:

- Klijent šalje paket s postavljenom SYN zastavicom poslužitelju kako bi pokrenuo vezu.

- Poslužitelj tada na taj početni paket odgovara s paketom s postavljenim SYN i ACK zastavicama kako bi potvrdio komunikaciju.
- Na kraju klijent vraća paket s postavljenom ACK zastavicom nakon čega je uspostavljena TCP veza i moguće je slati i primati podatke.

Napadač iskorištava činjenicu da nakon što je primljen početni paket za uspostavu veze, poslužitelj će odgovoriti s jednim ili više paketom i čekati posljednji korak u sinkronizaciji koji se neće u ni jednom trenutku dogoditi.

Tok napada izgleda sljedeće [5]:

- Napadač šalje veliki broj SYN paketa na ciljani poslužitelj s lažnim IP adresama.
- Poslužitelj odgovara na te zahtjeve za povezivanje i otvara jedan port očekujući odgovor.
- Veze na portovima koje očekuju zadnji korak TCP sinkronizacije ostaju poluotvorene i kada se zauzmu svi portovi poslužitelj više ne može normalno funkcionirati jer ne može uspostavljati nove veze.

Za ovaj napad postoje sljedeće protumjere [5]:

- Kako bi se obranili od ovog napada mogu se rezervirati dodatni memorijski resursi kako bi se biti povećao maksimalni broj mogućih poluotvorenih veza za rješavanje svih novih zahtjeva. Zbog rezerviranja dodatne memorije za obrađivanja veza može doći do pada performansi, ali taj ishod je i dalje bolji od slučaja gdje dolazi do prekida usluge.
- Kao protumjeru je isto moguće recikliranje najstarije poluotvorene veze nakon što su sve veze zauzete. Ova strategija zahtijeva da se legitimne veze mogu u potpunosti uspostaviti za manje vremena nego što se zaostatak može popuniti zlonamjernim SYN paketima [6]. Ova konkretna obrana ne uspijeva kada se poveća obujam napada ili ako je maksimalni broj mogućih veza premala.
- Moguća protumjera bi bila korištenje kolačića od strane poslužitelja. U ovom slučaju poslužitelj na svaki zahtjev za povezivanje odgovara SYN ACK paketom, ali zatim izbacuje SYN zahtjev za uspostavu veze iz reda čekanja ostavljajući port otvoren i spreman za novu vezu. Kada poslužitelj dobi ACK paket on će tada rekonstruirati (uz neka ograničenja) unos u red čekanja.
- Jedno od mogućih rješenja je i obrnuti posrednik [7] koji se nalazi između korisnika i sustava kojeg štiti koji će biti kasnije objašnjen u sljedećoj cjelini. U sklopu ovog rada je napravljena jednostavna verzija ovakvoga posrednika kao dokaz izvedivosti zaštite od tog napada na ovakav način.

2.2.2. ACK flood napad

ACK flood je napad uskraćivanja usluge 4. sloja (transportni sloj) i koristi TCP protokol kao i SYN flood napad. Napad je najefikasniji kada cilja uređaje koji održavaju nekakvo unutarnje stanje i trebaju obraditi svaki paket koji primi i zbog toga su sigurnosne stijene i poslužitelji pogodne mete za ACK flood napad [8]. Suprotno tome balanseri opterećenja, usmjerivači i prekidači nisu osjetljivi na ove napade. ACK poplavu je teško ublažiti iz nekoliko razloga. Može se lažirati pa napadač može lako generirati visoku stopu napadačkog prometa, a vrlo je

teško razlikovati legitimni ACK i napadački ACK paket, jer izgledaju isto prije nego što se obrade. Ovaj napad se isto može spriječiti obrnutim posrednikom jer on ne prosljeđuje nikakve ACK pakete koji nisu povezani s otvorenom TCP vezom. To osigurava da zlonamjerni ACK paketi ne stignu do izvornog poslužitelja.

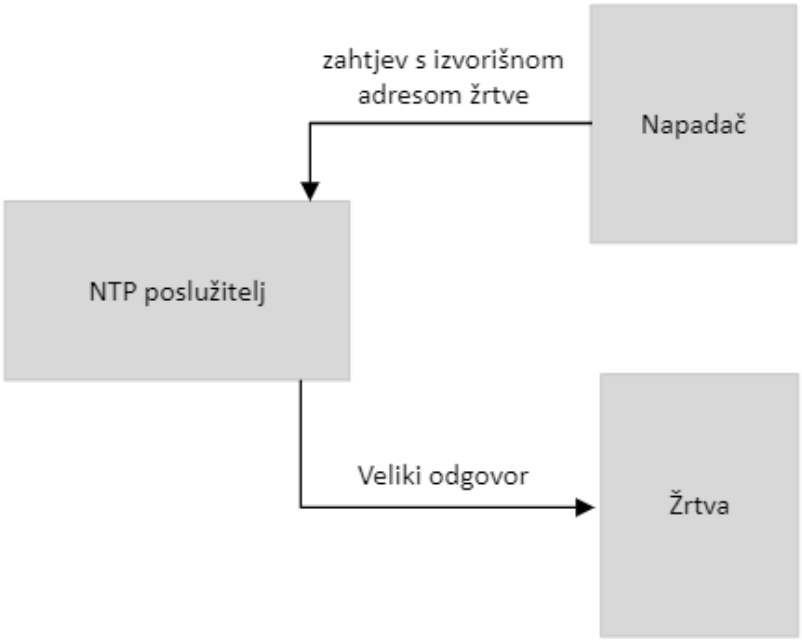
2.2.3. UDP flood napad

UDP flood je vrsta napada uskraćivanja usluga koji cilja mrežnu infrastrukturu preplavljivanjem ciljnog sustava ili mreže velikom količinom UDP paketa. UDP je protokol bez povezivanja koji se koristi za komunikaciju preko Interneta. U UDP flood napadu, napadač šalje veliki broj UDP paketa na određenu odredišnu IP adresu ili niz IP adresa. Cilj je potrošiti ciljne mrežne resurse, kao što su propusnost, procesorska snaga i memorija, kako bi sustav ili mreža postali neodgovarajući ili nedostupni legitimnim korisnicima. Napad radi iskorištavanjem prirode UDP protokola što veza nema statusa. Za razliku od TCP protokola, UDP ne uspostavlja vezu niti provjerava isporuku paketa. Stoga napadač može lako poslati veliku količinu UDP paketa s lažnom izvornom IP adresom, što otežava ući u trag podrijetlu napada.

Kada bi pretpostavili da meta napada treba samo TCP promet i dio DNS prometa (koji koristi UDP protokol ali se može filtrirati kako je opisano u poglavlju 2.2.5.) možemo jednostavno blokirati sav drugi ulazni UDP promet a da meta i dalje radi ispravno [9]. No moderni napadi velikog volumena mogu jednostavno preopteretiti sigurnosnu stijenu koja nije dizajnirana da obrađuje veliku količinu podataka [10].

2.2.4. NTP amplificirajući napad

Riječ je o napadu uskraćivanja usluga koji se temelji na refleksiji u kojem napadač zloupotrebljava funkcionalnost NTP poslužitelja kako bi preplavio ciljanu mrežu ili poslužitelj znatno uvećanom količinom UDP prometa. Amplificirajući napadi iskorištavaju nejednakost u omjeru stvorenog prometa između napadača i ciljane žrtve. Slanjem malih upita koji rezultiraju velikim odgovorima, zlonamjerni korisnik stvara znatno veći promet kao što je prikazano na slici 2.1.



Slika 2.1. Prikaz izvedbe NTP aplicirajućeg napada

NTP protokol omogućava uređajima povezanim s internetom da sinkroniziraju svoje interne satove. Iskorištavanjem naredbe poput monlist koja je omogućena na nekim NTP poslužiteljima, napadač može umnožiti svoj početni promet zahtjeva, što rezultira velikim odgovorom. NTP poslužitelj će na taj zahtjev odgovoriti sa 600 IP adresa koje su zadnje uputile zahtjev tom NTP poslužitelju i taj odgovor će onda biti 206 puta veći od početnog zahtjeva [11]. To znači da napadač s 1 GB internetskog prometa može izvesti napad od 200+ gigabajta što je ogroman porast stvorenog napadačkog prometa.

Budući da UDP zahtjevi koje šalje botnet moraju imati izvornu IP adresu lažiranu na IP adresu žrtve, ključna komponenta u smanjenju učinkovitosti amplificirajućih napada temeljenih na UDP-u je da davatelji internetskih usluga (*eng. ISP, Internet Service Provider*) odbiju svaki interni promet s lažne IP adrese [11]. Ako se paket šalje unutar mreže s izvornom adresom zbog koje izgleda kao da potječe izvan mreže, vjerojatno je riječ o lažnom paketu i može se odbaciti. Kombinacija onemogućavanja monlist-a na NTP poslužiteljima i implementacije ulaznog filtriranja na mrežama koje trenutno dopuštaju lažiranje IP adresa učinkovit je način zaustavljanja ove vrste napada prije nego što dosegne svoju namjeravanu mrežu.

2.2.5. DNS amplificirajući napad

DNS amplificirajući napad je sličan NTP-u. Tijekom napada, napadač iskorištava ranjivost u javno dostupnim DNS poslužiteljima kako bi preplavio metu velikim brojem UDP paketa. Kod ovog napada napadač isto koristi ugroženu krajnju točku za slanje UDP paketa s lažnim IP adresama, ali ovaj put prema DNS poslužitelju. Lažna adresa na paketima ukazuje na stvarnu IP adresu žrtve zbog čega će žrtva primiti odgovore od poslužitelja. Kako bi stvorio veliku količinu prometa, napadač isto strukturira zahtjev na način da generira što veći odgovor od DNS poslužitelja i zbog toga paketi često imaju argument „ANY“ [12]. Nakon što primi zahtjeve,

DNS šalje veliki broj UDP paketa na lažnu IP adresu koja pripada žrtvi koja prima odgovor i ta količina prometa ju preopterećuje, što rezultira uskraćivanjem usluge.

Jedno od rješenja za ovaj napad [13] nadzirao bi odlazne DNS zahtjeve od unutarnjeg ISP-a ili organizacije koju štiti i dolazne DNS odgovore uz pretpostavku da jedan dolazni odgovor odgovarao jednom odlaznom zahtjevu. Tijekom DNS amplificajućeg napada, ako su napadačeva računala i DNS poslužitelji izvan mreže koju štiti, ovo rješenje ne bi vidjelo zahtjeve prema DNS poslužiteljima, ali bi vidjela dolazne odgovore stoga bi broj odgovora bio veći od broja DNS zahtjeva tijekom napada. Slično rješenje bi isto bilo da se ne propuštaju odgovori DNS poslužitelja ako ne postoji zahtjev povezan s tim odgovorom [9].

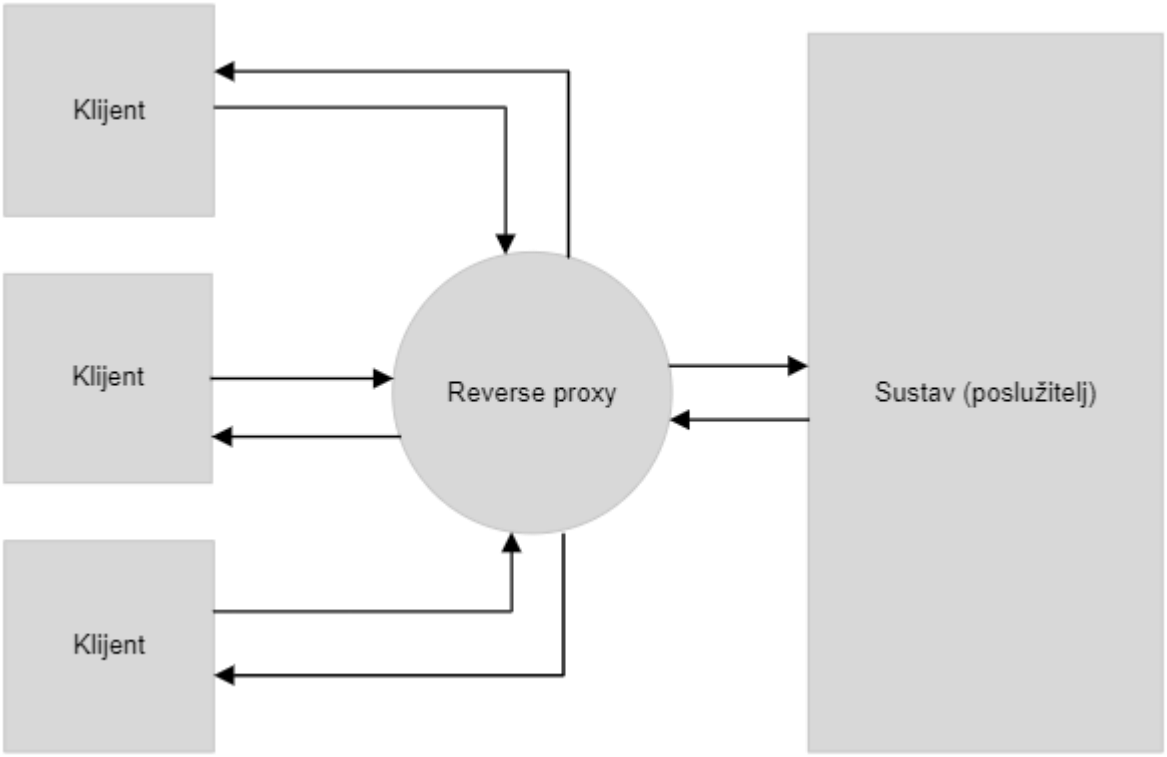
2.2.6. ICMP flood napad

ICMP (*eng. Internet Control Message Protocol*) flood napad je napad uskraćivanjem usluge u kojem napadač pokušava zatrpati ciljani uređaj paketima ICMP echo zahtjeva koji očekuju odgovor mete, s ciljem iskorištavanja ranjivosti u sposobnostima mete da obrađuje veliki promet ICMP paketa kako bi se poremetilo njeno normalno funkcioniranje ili u najgorem slučaju rezultiralo da meta postane nedostupna normalnom prometu. ICMP zahtjev zahtijeva resurse poslužitelja za obradu svakog zahtjeva i za slanje odgovora. Cilj ovog napada je nadvladati sposobnost ciljanog uređaja da odgovori na veliki broj zahtjeva i/ili preoptereti mrežnu vezu lažnim prometom. Budući da mnogi uređaji u botnetu ciljaju isto metu s ICMP zahtjevima, promet napada se značajno povećava, što potencijalno može rezultirati prekidom normalne mrežne aktivnosti.

Zaštita od ICMP preplavljivanja najjednostavnije je postići onemogućavanjem ICMP funkcionalnosti [9]. Žrtva u ovom slučaju i dalje radi ispravno, jedino kao posljedica imamo to da su sve mrežne aktivnosti koje uključuju ICMP onemogućene, zbog čega žrtva ne reagira na zahtjeve kao što su ping i traceroute koji koriste taj protokol.

3. Obrnuti posrednik

Jedno od mogućih rješenja je obrnuti posrednik [7][14] koji djeluje kao posrednik između korisnika i web poslužitelja, zamjenjujući IP adresu web poslužitelja vlastitom IP adresom. Pri pokretanju napada uskraćivanjem usluge na web poslužitelj, obrnuti posrednik će biti ciljan umjesto njega štiteći izvorni web poslužitelj od tog prometa. Kako bi se spriječilo da ovaj napad i dalje ometa peformanse poslužitelja, velike količine zahtjeva su raspodijeljeni kroz mrežu poslužiteljevih obrnutih posrednika koji apsorbiraju i smanjuju ukupni učinak napada. Obrnuti posrednik obrađuje sav promet između krajnjih korisnika i poslužitelja. Da bi se to postiglo, ova vrsta posrednika nalazi se na rubu mreže poslužitelja kao dodatna krajnja točka gdje prima sve početne zahtjeve za HTTP vezu prije nego što se pošalju izvornom poslužitelju. Kod obrnutog posrednika prikazanog na slici 3.1. kada klijenti šalju zahtjeve izvornom poslužitelju web-mjesta, te zahtjeve presreće obrnuti posrednik. Obrnuti posrednik će tada slati zahtjeve i primati odgovore od izvornog poslužitelja.



Slika 3.1. Prikaz obrnutog posrednika

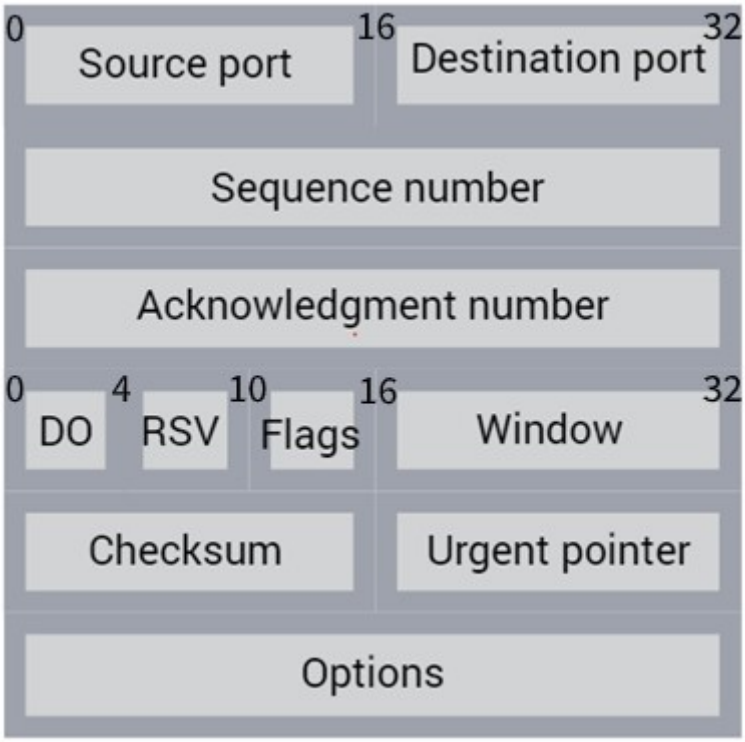
Pošto se u sklopu ovog rada traži zaštita koja ne prekida TLS tunel u ovom radu slijedi opis TCP i TLS protokola.

3.1. Protokol TCP

Jedan od protokola koji se zlouporabe u napadima uskraćivanja usluga je TCP (*eng. Transmission Control Protocol*). TCP jamči integritet podataka koji se prenose preko mreže između poslužitelja i klijenta. Prije nego što odašilje podatke, TCP uspostavlja vezu između izvora i odredišta, za koju osigurava da ostane živa dok komunikacija ne započne. Zatim razbija velike količine podataka u manje pakete, dok osigurava integritet podataka tijekom cijelog procesa.

Slika 3.2. prikazuje zaglavlje TCP paketa i polja koje se nalaze u njemu su

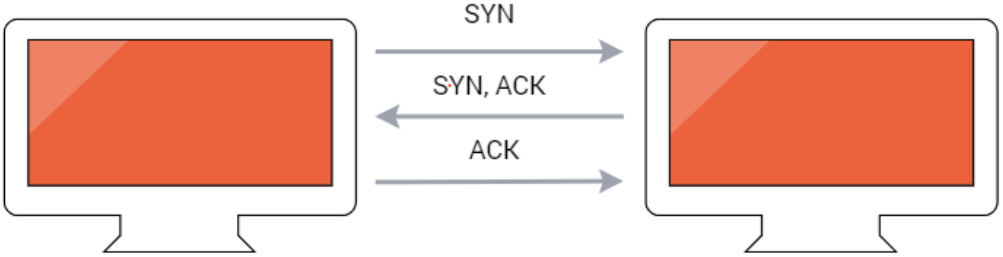
- Izvorni port: Predstavlja 16-bitno polje koje navodi broj porta pošiljatelja.
- Odredišni port: Predstavlja 16-bitno polje koje navodi broj porta primatelja.
- Redni broj: Označava 32-bitno polje koje pokazuje koliko je podataka poslano tijekom TCP sesije. Kada uspostavite novu TCP vezu tada je početni redni broj nasumična 32-bitna vrijednost. Primatelj će koristiti ovaj redni broj i poslati natrag potvrdu.
- Broj potvrde: Ovo 32-bitno polje primatelj koristi za traženje sljedećeg TCP segmenta. Ova vrijednost će biti redni broj uvećan za 1.
- Podatkovni pomak: Ovo je polje pomaka podataka od 4 bita, također poznato kao duljina zaglavlja. Označava duljinu TCP zaglavlja kako bi mogli znati gdje počinju stvarni podaci.
- Rezervirano polje: Predstavlja 3 bita za rezervirano polje. Oni su ne koriste nego su rezervirani za buduće korištenje i uvijek su postavljeni na 0.
- Zastavice: Označava 6 bitova za zastavice, nazivamo ih i kontrolnim bitovima. Koristimo ih za uspostavljanje veza, slanje podataka i prekid veza:
    - URG: hitan pokazivač. Kada je ovaj bit postavljen, podaci bi trebali imati prioritet u odnosu na druge podatke.
    - ACK: Zastavica koja se koristi za potvrdu.
    - PSH: Ova zastavica daje do znanja aplikaciji da se podaci trebaju poslati odmah i da ne želimo čekati da ispunimo cijeli TCP segment.
    - RST: Ovom zastavicom poništavamo vezu. Pri primitku poruke s ovom zastavicom veza se mora odmah prekinuti. Ovo nije normalan način na koji završava TCP veza i koristi se samo kada dođe do nepopravljive pogreške.
    - SYN: Ovu zastavicu koristimo kod početka sinkronizacije u tri koraka.
    - FIN: Ovaj završni bit se koristi za prekid TCP veze. TCP je dvosmjerna veza pa će obje strane morati koristiti FIN zastavicu za prekid veze. Ovo je normalna metoda prekida veze.
- Prozor: 16-bitno polje prozora određuje koliko bajtova je prijemnik spreman primiti. Koristi se kako bi primatelj mogao reći pošiljatelju da želi primiti više podataka od onoga što trenutno prima.
- Kontrolni zbroj: 16 bita se koristi za kontrolni zbroj za provjeru je li TCP zaglavlje ispravno ili ne.
- Hitni pokazivač: Ovih 16 bitova koristi se kada je postavljen URG bit, hitni pokazivač se koristi za označavanje gdje završavaju hitni podaci.
- Opcije: Ovo polje nije obavezno i veličina može biti bilo gdje između 0 i 320 bita.



Slika 3.2 zaglavlje protokola TCP

Da bi se uspostavila TCP veza izvršava se sinkronizacija u tri koraka preko sljedećih paketa gdje su podignute određene zastavice (slika 3.3):

- Klijent započinje vezu slanjem TCP paketa za podignutom SYN zastavicom svom odredištu. Paketi sadrže nasumični sekvencijski broj koji označava početak brojeva sekvence za podatke koje poslužitelj treba poslati.
- Poslužitelj će nakon toga primiti paket, a on će odgovoriti svojim sekvencijskim brojem. Njegov odgovor također uključuje zastavice SYN i ACK i broj potvrde, to jest redni broj klijenta uvećan za 1.
- Klijent odgovara poslužitelju slanjem broja potvrde koji je redni broj poslužitelja koji se povećava za 1 i podignutom zastavicom ACK.



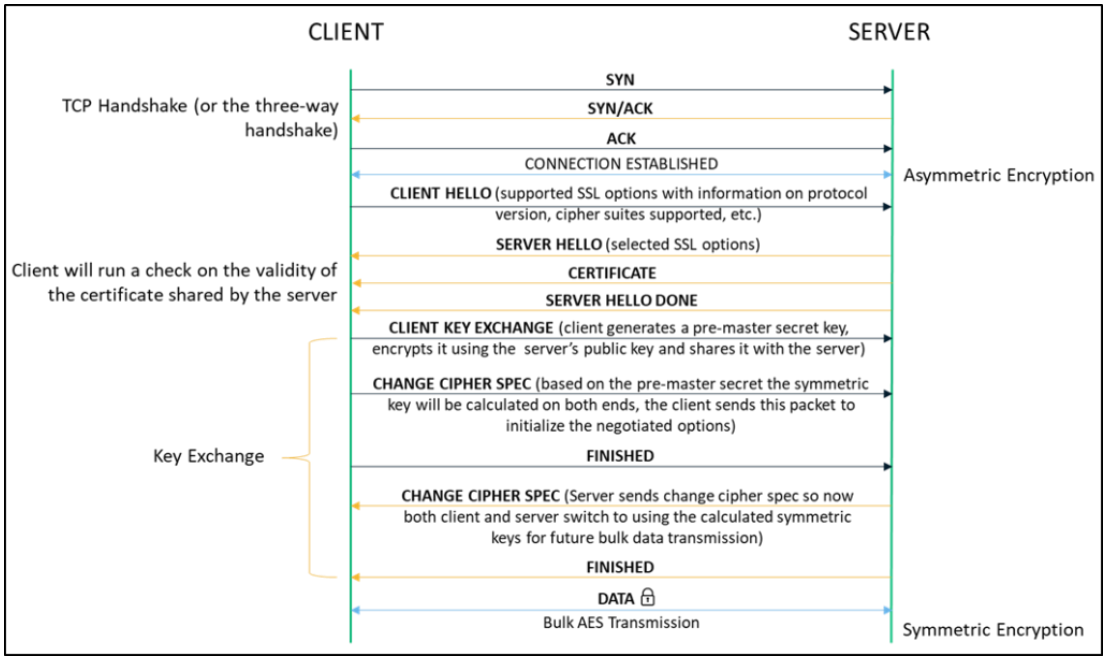
Slika 3.3 sinkronizacija u tri koraka protokola TCP

3.2. Protokol TLS

Pošto je PSD2 API zapravo REST (*eng. REpresentational State Transfer*) API on koristi HTTP (*eng. Hypertext Transfer Protocol*) kao svoj temeljni protokol koji donosi uobičajeni skup sigurnosnih problema. HTTPS (*eng. Hypertext Transfer Protocol Secure*) je proširenje HTTP protokola koji dodatno preko koristi TLS (*eng. Transport Layer Security*) protokol za enkripciju veze. Veza je kriptirana s zajedničkim simetričnim ključem koji se distribuira s javnim i privatnim ključem. TLS veza obično koristi HTTPS port 443. Alternativno, klijent također može poslati zahtjev poput STARTTLS za nadogradnju s nešifrirane veze na šifriranu.

Prije nego se uspostavi veza, preglednik i poslužitelj trebaju odlučiti o parametrima veze koji se mogu primijeniti tijekom komunikacije. Do dogovora dolaze izvođenjem SSL/TLS rukovanja (isto prikazan na slici 3.4.):

- Proces počinje razmjenom pozdravnih poruka između klijentskog preglednika i web poslužitelja (CLIENT HELLO i SERVER HELLO poruke) u tim porukama se isto komuniciraju standardi šifriranja koje podržavaju dvije strane.
- Poslužitelj zatim dijeli svoj certifikat.
- Klijent sada drži javni ključ poslužitelja, dobiven iz ovog certifikata. Provjerava valjanost certifikata poslužitelja prije upotrebe javnog ključa za generiranje predglavnog (eng. pre-master) tajnog ključa. Zatim se predglavna tajna šifrira javnim ključem i dijeli s poslužiteljem.
- Na temelju vrijednosti predglavnog tajnog ključa, obje strane neovisno izračunavaju simetrični ključ.
- Obje strane šalju poruku o promjeni specifikacije šifre koja pokazuje da su izračunale simetrični ključ i za daljnji prijenos podataka koristit će simetričnu enkripciju.



Slika 3.4. rukovanje protokola TLS

3.3. Implementacija posrednika

Svrha ovog posrednika je da primarno štiti od napada preplavljivanjem koji zloupotrebljavaju TCP protokol i posrednik će prosljeđivati samo komunikaciju koja koristi taj protokol. Ideja iza ovog posrednika je da poslužitelj ne ulazi u direktni kontakt sa potencijalnim klijentima koji neće uspostaviti kompletnu TCP vezu ili šalju pakete koji ne pripadaju uspostavljenoj TCP vezi. Umjesto da posrednik prosljeđuje inicijalni SYN paket od klijenata preko kojega on želi uspostaviti vezu s poslužiteljem, posrednik će prvo sam pokušati uspostaviti vezu sa poslužiteljem šaljući nazad TCP paket s postavljenim zastavicama SYN i ACK. Zbog ovoga poslužitelj ne mora rukovati poluotvorenim vezama. Ukoliko klijent završi treći korak TCP rukovanja s posrednikom, posrednik će jedino u tom slučaju uspostavi kompletnu TCP vezu s poslužiteljem. Nakon što posrednik uspostavi vezu s poslužiteljem posrednik prosljeđuje pakete koje dobiva s veze uspostavljene s klijentom prema poslužitelju i obratno.

Posrednik je napisan u programskom jeziku python i koristi biblioteke socket i socketserver. U implementaciji za posrednika koristimo klasu PSD2Proxy prikazana u isječku 3.1 koji koristi TCPServer klasu socketserver biblioteke i pod argumente još navodimo ThreadingMixIn jer želimo da sve zahtjeve obrađujemo u novim dretvama.

```
class PSD2Proxy(socketserver.ThreadingMixIn,socketserver.TCPServer):
    pass
```

Ispis 3.1 Dio koda koji definira klasu PSD2Proxy

Klasa za rukovanje zahtjevima ThreadedTCPRequestHandler prikazana je u isječku 3.2 i mora nadjačati metodu handle() za implementaciju komunikacije s klijentom. Pri instanciranju klase PSD2Proxy ThreadedTCPRequestHandler predaje se našem posredniku kao argument kako je prikazano u isječku 3.3. Posrednik sluša na portu 443 i kada se izvrši TCP sinkronizacija u tri koraka poziva se metoda hande() klase ThreadedTCPRequestHandler koja zatim uspostavlja novu pristupnu točku (eng. socket) s bankom i tada koristeći dvije zasebne dretve prosljeđuje podatke s jedne pristupne točke na drugu uključujući i dogovaranje TLS veze. Zbog toga posrednik ni u jednom trenutku nema uvid na osjetljive podatke u čitljivom formatu niti posrednik treba imati ikakve certifikate u ovom procesu jer posrednik ne sudjeluje u dogovaranju TLS veze nego ju on samo prosljeđuje.

```
class ThreadedTCPRequestHandler(socketserver.BaseRequestHandler):

     def handle(self):
          print("Starting to handle connection...")
          f = Forwarder(self)
          f.start()
          try:
               while True:
               data = self.request.recv(4096)
               if len(data) == 0:
                    raise Exception("endpoint closed")
               print("Received from source: " + str(len(data)))
               f.write_to_dest(data)
```

```
            except Exception as e:
                 print("EXCEPTION reading from main socket")
                 print(e)
            f.stop_forwarding()
            print("...finishing handling connection")
```

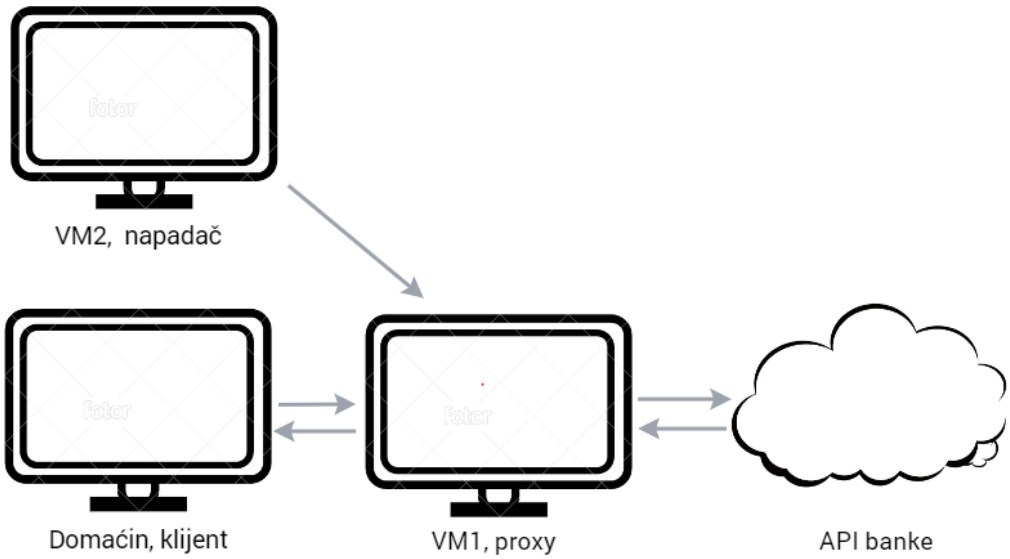
Ispis 3.2 Kod klase ThreadedTCPRequestHandler sa metodom handle()

```
server = PSD2Proxy((HOST, PORT), ThreadedTCPRequestHandler)
server_thread = threading.Thread(target=server.serve_forever)
server_thread.daemon = True
server_thread.start()
```

Ispis 3.3 Instanciranje klase PSD2Proxy

4. Testiranje

Za testiranje napada koristiti će se dvije virtualne mašine i njihov domaćin (slika 4.1.). Pri testiranju napada posrednikova skripta se pokreće sa virtualne mašine VM1 koja ima dva mrežna sučelja. Prvo sučelje je host-only mrežno sučelje koje povezuje virtualnu mašinu VM1 sa domaćinom i virtualnom mašinom VM2. S tog mrežnog sučelja će dolaziti paketi koje šalje skripta za napade koja se nalazi na virtualnoj mašini VM2 i osim testiranja napada, da bi se prikazalo ispravna obrada legitimnih korisničkih zahtjeva napravljene je klijentska skripta koja se nalazi na domaćinu i šalje zahtjeve prema Sandbox verziji OPENAPI-ja Erste banke [15]. Drugo mrežno sučelje je NAT sučelje i preko njega možemo gledati promet koji posrednik prosljeđuje dalje prema poslužitelju, u ovom slučaju API-ju banke.



Slika 4.1. Prikaz okruženja za testiranje

4.1. Testiranje klijentske skripte

Za svrhu ovog rada napravljena je jednostavna klijentska skripta koja prati jedan tok u kojemu dobiva pristupni token (*eng. Access token*) nakon kojega će se uputiti zahtjev za informacije o računima u Sandbox okruženju. Klijentska skripta se pokreće na operacijskom sustavu domaćina i promet klijentske skripte se preusmjerava na adresu virtualne mašine od host-only mrežnog sučelja.

Da bi se moglo pristupiti API-ju banke potrebno je prvo registrirati aplikaciju na portalu Erste banke. Kada je ovaj korak dovršen, moguće je generirati vjerodajnice za Sandbox okruženje. Nakon uspješnog generiranja vjerodajnica, zajedno s osnovnim URL-om za Sandbox API, imamo sve podatke potrebne za izvođenje poziva Sandbox API-ja a to su osnovni URL za API, API ključ, ID klijenta i klijentova tajna. Sve informacije o krajnjim točkama mogu se pronaći

u API dokumentaciji [16]. Sve krajnje točke zahtijevaju dvosmjerni TLS i koriste QWAC certifikat za pozivanje krajnje točke. Testni QWAC certifikati dostupni su za preuzimanje s Erste Developer Portala na stranici unutar detalja registrirane aplikacije. Nakon što dobijemo certifikat za klijenta da bi dobili pristupni token (eng. Access token) koristimo SCA (*eng. Strong Customer Authentication*) tok koji započinjemo pozivanjem krajnje točke POST /consents kako bi stvorili consent objekt i dobili Consent-ID. Taj zahtjev nam vraća JSON oblika prikazan na ispisu 4.1.

```
{
    "consentStatus": "received",
    "consentId": "cfca80e8-63c7-4c27-97b4-9277447e4055",
    "_links": {
        "self": {
            "href": "/consents/cfca80e8-63c7-4c27-97b4-
9277447e4055"
        },
        "status": {
            "href": "/consents/cfca80e8-63c7-4c27-97b4-
9277447e4055/status"
        },
        "scaStatus": {
            "href": "/consents/cfca80e8-63c7-4c27-97b4-
9277447e4055/authorizations/20e50391-d9c9-4306-88b7-
6dc849d05068"
        }
    }
}
```

Ispis 4.1 Odgovor na zahtjev POST /consents

Preko krajnje točke GET /consents/e9461c04-d7e0-4fd3-88983e4250f75e10/authorizations/20e50391-d9c9-4306-88b7-6dc849d05068 povremeno provjeravamo status autentifikacije dok ne dobijemo povratnu informaciju da je autorizacija finalizirana od strane korisnika dobivanjem statusa "finalised" kako je prikazano na ispisu 4.2.

```
{
  "scaStatus": "finalised"
}
```

Ispis 4.2 Odgovor na zahtjev za statusom autorizacijskog procesa

Kada dobijemo status finalised možemo nastaviti sa stjecanjem tokena pozivanjem krajnje točke s istim URL-om kao kod prošlog zahtjeva uz dodatak "/token" pa bi nam krajnja točka bila POST "/consents/cfca80e8-63c7-4c27-97b4-9277447e4055/authorizations/20e50391-d9c9-4306-88b7-6dc849d05068/token" gdje dobivamo odgovor prikazan u ispisu 4.3.

```
{
```

```
  "access_token":
"ewogICJ0eXBlIjogImFjY2VzcyIsCiAgImV4cGlyYXRpb24iOiAiMjAyMy0wN
i0xNVQxMjoyMzowOC4xNTdaIiwKICAidXNlcklkIjogImM5ZjBkYzM2LWQ4YmQ
tNGU2MC05MTVhLWFkNzgzNGM5MzM3MiIsCiAgInNlc3Npb25VVUlEIjogImMxM
2YzZThmLWQ0Y2UtNDlmZC1iNjJkLTE5YmEzZmM0NTYyNSIsCiAgInNjb3BlcyI
6IFsKICAgICJhaXM6Y2ZjYTgwZTgtNjNjNy00YzI3LTk3YjQtOTI3NzQ0N2U0M
DU1IiwKICAgICJvcGVuaWQiLAogICAgIm9mZmxpbmVfYWNjZXNzIgogIF0sCiA
gImNvbnNlbnRJZCI6ICJjZmNhODBlOC02M2M3LTRjMjctOTdiNC05Mjc3NDQ3Z
TQwNTUiLAogICJsaW1pdHMiOiB7CiAgICAiYWNjZXNzU2Vjb25kcyI6IDE4MDA
sCiAgICAicmVmcmVzaFNlY29uZHMiOiAxNzI4MDAwCiAgfSwKICAiYWNjZXNzV
HlwZSI6ICJvZmZsaW5lIgp9",
  "token_type": "bearer",
  "expires_in": 1800,
  "scope": "ais: cfca80e8-63c7-4c27-97b4-9277447e4055 openid
offline_access",
  "refresh_token":
"ewogICJ0eXBlIjogInJlZnJlc2giLAogICJleHBpcmF0aW9uIjogIjIwMjMtM
DctMDVUMTE6NTM6MDguMTcwWiIsCiAgInVzZXJJZCI6ICJjOWYwZGMzNi1kOGJ
kLTRlNjAtOTE1YS1hZDc4MzRjOTMzNzIiLAogICJzZXNzaW9uVVVJRCI6ICJjM
TNmM2U4Zi1kNGNlLTQ5ZmQtYjYyZC0xOWJhM2ZjNDU2MjUiLAogICJzY29wZXM
iOiBbCiAgICAiYWlzOmNmY2E4MGU4LTYzYzctNGMyNy05N2I0LTkyNzc0NDdlN
DA1NSIsCiAgICAib3BlbmlkIiwKICAgICJvZmZsaW5lX2FjY2VzcyIKICBdLAo
gICJjb25zZW50SWQiOiAiY2ZjYTgwZTgtNjNjNy00YzI3LTk3YjQtOTI3NzQ0N
2U0MDU1IiwKICAibGltaXRzIjogewogICAgImFjY2Vzc1NlY29uZHMiOiAxODA
wLAogICAgInJlZnJlc2hTZWNvbmRzIjogMTcyODAwMAogIH0sCiAgImFjY2Vzc
1R5cGUiOiAib2ZmbGluZSIKfQ=="
}
```


Ispis 4.3 Odgovor na zahtjev koji vraća pristupni token

S dobivenim tokenom onda pozivamo krajnju tuču GET /accounts i dobivamo sljedeći prikaz stanja računa kako je prikazano na ispisu 4.4.


```
{
  "accounts": [
    {
      "resourceId": "6babfaf3-6239-44a1-a32a-7482f95f77a5",
      "iban": "HR6924020063209999998",
      "currency": "EUR",
      "name": "",
      "product": "",
      "status": "enabled",
      "usage": "PRIV",
      "balances": [
        {
          "balanceAmount": {
            "amount": 1400,
            "currency": "EUR"
```

```
          },
          "balanceType": "interimAvailable",
          "referenceDate": "2017-12-20",
          "creditLimitIncluded": true,
          "lastChangeDateTime": "2017-11-21T14:18:19Z",
          "lastCommittedTransaction": "46356354624"
        }
      ],
      "_links": {
        "detail": {
          "href": "/accounts/6babfaf3-6239-44a1-a32a-
7482f95f77a5",
          "type": "GET"
        },
        "balances": {
          "href": "/accounts/6babfaf3-6239-44a1-a32a-
7482f95f77a5/balances",
          "type": "GET"
        },
        "transactions": {
          "href": "/accounts/6babfaf3-6239-44a1-a32a-
7482f95f77a5/transactions",
          "type": "GET"
        }
      },
      "ownerName": "PSU 1"
    }
  ]
}
```


Ispis 4.4 Odgovor na zahtjev GET /accounts

Pri izvođenu ove provjere posrednika koristeći alat Wireshark možemo vidjeti da posrednik uspješno uspostavlja vezu s legitimnim klijentom i razmjenjuje pakete.

4.2. Testiranje napada

Za testiranje napada se koristila MHDDoS skripta [17] s gotovim napadima koja se pokretala s druge virtualne mašine s operacijskim sustavom Kali Linux koji dolazi sa hping3 alatom. MDDDoS skripta nudi veliki broj napada na transportnom i aplikacijskom sloju osi modela ali zbog poteškoća pri pokretanju nekih napada skripte domaćinovog računala čiji je operacijski sustav Windows skripta se pokreće s Kali Linux virtualne mašine VM2. Svi testirani napadi su na razini transportnog sloja od MHDDoS skripte se pokreću naredbom prikazanom na ispisu 4.5. sa sljedećim argumentima:

- Method: Vrsta napada želimo pokrenuti.
- Ip:port: URL ili IP adresa sa portom žrtve koju napadamo, u ovom slučaju to će biti IP adresa virtualne mašine VM1 (Slika 4.1).
- Threads: Broj dretvi koje će se koristiti pri napadu.
- Duration: Trajanje napada izraženo u sekundama.
- Debug: Opcionalni argument i ukoliko je odabran pri napadu će se u komandnoj liniji ispisivati status napada tokom njegovog izvođenja.

```
python3 start.py <1=method> <2=ip:port> <3=threads> <4=duration>
<5=debug=optional>
```

Ispis 4.5 Opis naredbe za pokretanje napada MHDDoS skriptom

Pri izvođenju napada on se nadgledao tako da se pomoću alata Wireshark pratio promet na dvjema mrežnih kartica. Na host-only mrežnoj kartici preko koje posrednik prima pakete od klijentske skripte i pakete od napada koje testiramo i na NAT mrežnoj kartici preko koje posrednik pristupa internetu i prosljeđuje zahtjeve prema API-ju banke.

4.2.1. SYN flood napad

MHDDoS skripta ima implementirani SYN flood napad gdje skripta otvara više konekcija šaljući pakete sa postavljenom zastavicom SYN i taj napad se pokreće u komandnoj liniji naredbom u primjeru 4.6.

```
python3 start.py syn 192.168.56.102:443 1 2 true
```

Ispis 4.6 Primjer pokretanja SYN flood napada MHDDoS skriptom

SYN flood napad se isto može pokrenuti pomoću alata hping3 [18] koristeći naredbu u komandnoj liniji prikazanu u primjeru 4.7.

```
hping3 -c 500 -d 120 -S -w 64 -p 443 --fast --rand-source
192.168.56.102
```

Ispis 4.7 Primjer pokretanja SYN flood napada alatom hping3

Gore navedenom naredbom smo definirali broj paketa koji je 500, veličine 120 bajta s veličinom TCP prozora 64 koji idu na adresu virtualne mašine VM1 s portom 443 na kojoj se nalazi posrednik. Ti paketi imaju postavljenu SYN zastavicu s opcijom -S sa slučajnom izvornom adresom zbog opcije --rand-source. Intenzitet napada se isto može definirati vlastitim definiranjem intervala slanja paketa u mikrosekundama navođenjem željenog intervala nakon opcije -i. Umjesto definiranja intenziteta isto je moguć odabir već definirane brzine gdje recimo opcijom --fast postavljamo interval na 10000 mikrosekundi, opcijom --faster postavljamo

interval između paketa na 1 mikrosekundu i konačno opcijom --flood se paketi šalju što brže moguće.

Pri pokretanju napada a host-only mrežnoj kartici se vide TCP paketi koji imaju postavljenu zastavicu SYN. Posrednik ne prosljeđuje nikakve pakete prema poslužitelju i na NAT mrežnoj kartici se mogu vidjeti paketi od posrednika sa postavljenom zastavicom SYN i ACK jer posrednik pokušava uspostaviti vezu na lažiranim izvorišnim adresama. Isto tako pri izvođenju ove provjere posrednika možemo vidjeti da se nikada ne završi TCP sinkronizacija u tri koraka kod nelegitimnih paketa i napad je zaustavljen kod posrednika i ne širi se dalje prema sustavu kojega štiti.

4.2.2. ACK flood napad

Pošto MHDDoS skripta nema ponuđen ovaj napad ACK flood napad se može pokrenuti pomoću alata hping3 preko naredbe u komandnoj liniji prikazanu u primjeru 4.8. Pokretanje hping3 alata je slično kao kod SYN flood napada, jedino se kod paketa dodaje opcija -A kako bi ti paketi imali postavljenu ACK zastavicu.

```
hping3 -c 500 -d 120 -A -w 64 -p 443 --fast --rand-source
192.168.56.102
```

Ispis 4.8 Primjer pokretanja ACK flood napada alatom hping3

Pri izvođenju napada kada gledamo promet na mrežnim sučeljima vidimo da prema posredniku dolaze TCP paketi s postavljenim ACK zastavicama na koje posrednik odgovara sa TCP paketima s postavljenom RST zastavicom koja javlja da primjeni TCP paketi pripadaju neispravnim vezama te da se ta veza mora prekinuti.

4.2.3. UDP flood napad

MHDDoS skripta ima implementirani UDP flood napad gdje skripta šalje veliki broj UDP paketa. Napad je bio pokrenut u komandnoj liniji kako je prikazano u primjeru 4.9.

```
python3 start.py udp 192.168.56.102:443 1 2 true
```

Ispis 4.9 Primjer pokretanja UDP flood napada MHDDoS skriptom

Kod ovog napada posrednik ne prosljeđuje nikakve pakete prema API-ju banke, a na NAT mrežnoj kartici se mogu vidjeti paketi kako posrednik odgovara na te pakete tako da šalje ICMP pakete s porukom da je port nedostupan. Ovaj napad je isto moguće izvesti koristeći hping3 alat naredbom u komandnoj liniji kako je prikazano u primjeru 4.10.

```
sudo hping3 --udp -port 443 --fast --rand-source 192.168.56.102
```

Ispis 4.10 Primjer pokretanja UDP flood napada alatom hping3

4.2.4. ICMP flood napad

MHDDoS skripta ima implementirani ICMP flood napad gdje skripta šalje veliki broj ICMP echo request paketa.

Napad je bio pokrenut u komandnoj liniji kako je prikazano u primjeru 4.1.

```
python3 start.py icmp 192.168.56.102:443 1 2 true
```

Ispis 4.11 Primjer pokretanja ICMP flood napada MHDDoS skriptom

Pri pokretanju ovog napada na host-only mrežnoj kartici se vide UDP paketi. Kod ovog napada posrednik ne prosljeđuje nikakve pakete prema API-ju banke, a na NAT mrežnoj kartici se mogu vidjeti paketi kako posrednik odgovara na te pakete sa ICMP ehco reply porukom. Ovaj napad je isto moguće izvesti koristeći hping3 alat naredbom u komandnoj liniji kako je prikazano u primjeru 4.12.

```
sudo hping3 --icmp -port 443 --fast --rand-source 192.168.56.102
```

Ispis 4.12 Primjer pokretanja ICMP flood napada alatom hping3

5. Zaključak

Ovim radom su pregledane moguće zaštite od napada uskraćivanja usluga i uspješno demonstrirano rješenje gdje se koristi obrnuti posrednik koji bi štitio sustav od napada preplavljivanjem koji koriste različite protokole. U demonstraciji intenzitet napada nije bio ni približno velik u usporedbi sa stvarnim napadima, no cilj ovog rada je bio dokazati izvedivost ovog rješenja bez prekidanja TCP tunela gdje se za svrhu testiranja koristila MHDDoS skripta s metodama za napade uskraćivanja usluga i alat hping3. Također kako bi se dokazalo da je moguće da posrednik prosljeđuje regularan promet za tu je svrhu napravljena klijentska skripta koja šalje zahtjeve prema API-ju Erste banke. Iako je unutar ovog rada posrednik bio originalno namijenjen kao protumjera kod SYN flood napada gdje posrednik obrađuje zahtjeve za TCP vezu, tako da stvarna meta ne mora rukovati poluotvorenim vezama, posrednik po svojoj prirodi može štiti i od drugih napada preplavljivanja koji su bili testirani u sklopu ovoga rada.

6. Literatura

[1] Nazrul Hoque, Dhruba K. Bhattacharyya, Jugal K. Kalit. „Botnet in DDoS Attacks: Trends and Challenges“ IEEE Communications Surveys & Tutorials (Volume: 17, Issue: 4, Fourthquarter 2015)

[2] Oliver Adam, SSL DDoS Attacks and How to Defend Against Them, link11, (2018. kolovoz), Poveznica: https://www.link11.com/en/blog/threat-landscape/ssl-ddos-attacks-and-how-to-defend-against-them/; pristupljeno 25. lipnja 2023.

[3] What is a CDN? | How do CDNs work?, Cloudflare, Poveznica: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/; pristupljeno 25. lipnja 2023.

[4] Amit Praseed, P. Santhi Thilagam, „DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications“, (2019. kolovoz)

[5] SYN flood attack, Cloudflare, Poveznica: https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/; pristupljeno 25. lipnja 2023.

[6] Rocky K. C. Chang, Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial, The Hong Kong Polytechnic University (2002. listopad)

[7] What is a reverse proxy? | Proxy servers explained, Cloudflare,

Poveznica: https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/; pristupljeno 25. lipnja 2023.

[8] ACK flood, Red Button,

Poveznica: https://www.red-button.net/ddos-glossary/ack-flood/; pristupljeno 25. lipnja 2023.

[9] Jelena Mirkovic, Peter Reiher, „A Taxonomy of DDoS Attack and DDoS Defense Mechanisms“ (2004. travanj)

[10] UDP flood attack, Cloudflare, Poveznica: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/ ; pristupljeno 25. lipnja 2023.

[11] NTP amplification DDoS attack, Cloudflare, Poveznica: https://www.cloudflare.com/learning/ddos/ntp-amplification-ddos-attack/; pristupljeno 25. lipnja 2023.

[12] DNS amplification attack, Cloudflare, Poveznica: https://www.cloudflare.com/learning/ddos/dns-amplification-ddos-attack/; pristupljeno 25. lipnja 2023.

[13] Changhua Sun, Bin Liu and Lei Shi , „Efficient and Low-Cost Hardware Defense Against DNS Amplification Attacks“ (2008.)

[14] Proxy Firewall, Fortinet, Poveznica: https://www.fortinet.com/resources/cyberglossary/proxy-firewall; pristupljeno 25. lipnja 2023.

[15] Getting started, Erste Group, Poveznica: https://developers.erstegroup.com/docs/guides/general-getting-started; pristupljeno 25. lipnja 2023.

[16] PSD2 Account Information API Documentation, Erste Group, Poveznica: https://developers.erstegroup.com/docs/apis/bank.ebc/bank.ebc.v1%2Fpsd2-aisp; pristupljeno 25. lipnja 2023.

[17] MHDDoS - DDoS Attack Script With 56 Methods, Github, Poveznica: https://github.com/MatrixTM/MHDDoS; pristupljeno 25. lipnja 2023.

[18] hping3(8) - Linux man page, Die.net, Poveznica: https://linux.die.net/man/8/hping3; pristupljeno 25. lipnja 2023.

Naslov, sažetak i ključne riječi

Naslov: Napadi uskraćivanja usluge na PSD2 API i zaštita bez prekidanja TLS tunela

Ovaj diplomski rad iznosi pregled i moguće zaštite od napada uskraćivanja usluga od kojih se je moguće zaštititi bez prekidanja TLS tunela. Napadi su podijeljeni na napade na razini transportnog i napade na razini aplikacijskog sloja OSI modela.

Unutar ovog rada se koristio jednostavni obrnuti posrednik pomoću kojega se moglo pokazati izvedivost obrane od ovakvog tipa napada bez TLS tunel prekinuo. Pri testiranju posrednika za slanje legitimnog prometa napravljena skripta koja šalje zahtjeve prema Sanbox verziji OPENAPI-ja Erste banke a za testiranje napada se koristila MHDDOS skripta i hping3 alat.

Ključne riječi: informacijska i kibernetička sigurnost, PSD2, reverse proxy, proxy, DDoS, Content Delivery Network, sigurnosna stijena

Title, summary and keywords

Title: Denial of Service Attacks on PSD2 API and Protections Without Terminating TLS Tunnel

This thesis presents an overview and possible defenses against denial-of-service attacks that can be protected against without breaking the TLS tunnel. Attacks are divided into attacks at the transport level and attacks at the application layer level of the OSI model.

Within this work, a simple reverse proxy was used to demonstrate the feasibility of defending against this type of attack without the TLS tunnel being broken. When testing the intermediary for sending legitimate traffic, a script was created that sends requests to the Sanbox version of Erste Bank's OPENAPI, and the MHDDOS script and the hping3 tool were used to launch the attacks.

Keywords: information and cyber security, PSD2, reverse proxy, proxy, DDoS, Content Delivery Network, Firewall