Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Vesna Glavač

Možnosti zagotavljanja anonimnosti

diplomsko delo

univerzitetni interdisciplinarni študijski program prve stopnje računalništvo in matematika

Mentorica: doc. dr. Mojca Ciglarič

Ljubljana 2012

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00013/2012
Datum: 12.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdaja naslednjo nalogo:

Kandidat: **VESNA GLAVAČ**

Naslov: **MOŽNOSTI ZAGOTAVLJANJA ANONIMNOSTI**
DIFFERENT WAYS OF PROVIDING ANONYMITY

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Preučite področje anonimnosti v porazdeljenih sistemih. Navedite različne vrste anonimnosti, pojasnite anonimnost omrežja in razložite, kam v komunikacijske plasti po ISO OSI modelu se uvršča zagotavljanje anonimnosti. Nato preučite, kako anonimnost zagotavlja protokol SOCKS, poiščite njegove prednosti in slabosti in predlagajte, kako bi lahko trenutnim funkcionalnostim dodali tudi zaupnost komunikacije. Protokol preizkusite v praksi na testnem poligonu in kritično ovrednostite možnosti njegove uporabe.

Mentor:

M. Ciglarič

doc. dr. Mojca Ciglarič

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Nikolaj Zimic

Dekan Fakultete za matematiko in fiziko:

akad. prof. dr. Franc Forstnerič

za

Izjava o avtorstvu diplomskega dela

Spodaj podpisana Vesna Glavač, z vpisno številko **63080040**, sem avtorica diplomskega dela z naslovom:

Možnosti zagotavljanja anonimnosti

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom doc. dr. Mojce Ciglarič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 21. septembra 2012 Podpis avtorja:

Zahvala

Zahvaljujem se mentorici doc. dr. Mojci Ciglarič za pomoč pri izdelavi diplomskega dela. Zahvale gredo Davidu Moharju, ki mi je pomagal pri študiju in nabiranju izkušenj na področju računalništva in Roku Sokolu za vso pomoč in podporo. Zahvaljujem pa se predvsem družini, ki mi je vsa leta študija stala ob strani in mi pomagala z nasveti, potrpežljivostjo in vztrajnostjo. Posebej bi se še rada zahvalila staršem za psihično in finančno podporo.

Svojim dragim staršem in Davidu.

Kazalo

Povzetek

Cilj diplomske naloge je opozoriti na pomen varnosti in anonimnosti na internetu s prikazom primera uporabe posredovalniškega strežnika SOCKS. V teoretičnem delu so opisane značilnosti različnih varnostnih mehanizmov, ki se v praksi najpogosteje uporabljajo. Poudarek je na protokolu SOCKS, ker se poleg posredovalniškega strežnika HTTP največ uporablja. Predstavljeno je delovanje, uporaba, prednosti, slabosti ter verzije protokola SOCKS. Opisan je tudi model ISO/OSI, ki je po plasteh razčlenjen glede na zagotavljanje anonimnosti. Bistvo je bilo najti čim bolj enostaven način uporabe protokola SOCKS za kriptiranje prometa z uporabo tunela SSH. Izkazalo se je, da je mogoče z enim ukazom postaviti posredovalniški strežnik SOCKS5 in tunel SSH hkrati. S tem zagotovimo poleg varnosti še kriptiranje prometa.

Ključne besede

protokol SOCKS, anonimnost, posredovalniški strežnik, čebulasto usmerjanje, varnost, model ISO/OSI

Abstract

The aim of this thesis is to show the importance of security and anonymity on the internet by demonstrating the use of a SOCKS proxy. In theoretical part, properties of various security mechanisms are explored with focus on SOCKS protocol as one of the most widely used systems. Presented are the inner workings and different versions as well as advantages and drawbacks of the protocol. The paper tries to identify the simplest way of encrypting data traffic by using SOCKS protocol with SSH tunneling. It will be demonstrated that it is possible to set up a SOCKS5 proxy server with SSH tunneling by use of a single command, thus ensuring the integrity of data.

Keywords

SOCKS protocol, anonymity, proxy server, onion routing, security, ISO/OSI model

Poglavje 1

Uvod

Vsak dan uporabljamo računalnike in internet. Srečali smo se že z mnogimi aplikacijami in programi. Vemo, kako izgledajo, kaj omogočajo, kako se uporabljajo, dostikrat pa nismo pozorni na varnost in anonimnost v ozadju. Količina informacij se eksponentno povečuje, s tem pa tudi potreba po dodatnih varnostnih mehanizmih. Z obsegom informacij se povečuje stopnja spletnega kriminala, s tem pa tudi potreba po razvoju dodatnih varnostnih mehanizmov. Skrbniki aplikacij in programov vedno bolj opozarjajo na to problematiko. Potreba po večjem računalniškem opismenjevanju in varnosti na internetu je tudi ena izmed motivacij za to diplomsko nalogo.

Število uporabnikov interneta se je povečalo tudi pri starejših generacijah, vendar je pri njih najbolj prisoten strah oziroma imajo največ vprašanj glede varne uporabe interneta in kako se zavarovati pred zlorabami. Naloga računalniško izobraženih je med drugim tudi računalniško opismenjevanje ostalih uporabnikov. S tem bomo omogočili razvoj interneta, hkrati moramo poskrbeti tudi za ustrezen razvoj varnostnih mehanizmov.

V nalogi je opisanih več možnih varnostnih mehanizmov, ki jih uporabljamo, kot so na primer čebulasto usmerjanje, slepo opuščanje, posredovalniški strežniki in drugi. Poleg anonimnosti je poudarek na protokolu SOCKS, saj je eden izmed najboljših mehanizmov za zagotavljanje anonimnosti in varnosti na internetu. Z razvojem protokola SOCKS5, se njegova uporaba razširja.

V praktičnem delu naloge smo prikazali pogost primer uporabe protokola SOCKS5 z uporabo protokola SSH in varnega tunela.

Poglavje 2

Anonimnost

Anonimnost izvira iz grške besede *anonymia*, kar pomeni "brez imena", "brezimnost". V pogovorni rabi se anonimnost tipično nanaša na stanje posameznikove identitete, ki javno ni znana. V računalništvu si anonimnosti želi vsakdo, ki se boji maščevanja, osramotitve, nepopularnih čustev ali si preprosto želi zasebnosti na internetu.

Uporablja se v podjetjih za prijavo nepravilnosti, v organih pregona za anonimne namige o kaznivih dejanjih, na univerzah za ocenjevanje poteka predmeta, v vladi za politične diskusije in volitve, v restavracijah za povratne informacije strank, v revijah, časopisih in raznih ostalih (internetnih) svetovalnicah za podajanje nasvetov bralcem in še veliko ostalih načinov.

Anonimnost velikokrat zamešamo s pojmom zasebnosti. Imata podoben koncept, a vendar sta si različna. Zasebnost omrežja preprečuje opazovalcu vpogled v informacije, ki se prenašajo po omrežju. Zasebnost omrežja ponavadi vključuje sisteme kodiranja, kriptografijo in/ali steganografijo. Anonimnost omrežja preprečuje identifikacijo bodisi pošiljatelja bodisi prejemnika oziroma prejemnike. Primeri protokolov, ki zagotavljajo zasebnost omrežja, a ne omogočajo anonimnosti so SSH, SSL in IPsec. Obratno, veliko omrežnih sistemov, ki omogočajo anonimnost, ne zagotavlja zasebnosti, na primer posredovalniški strežniki in sistemi slepega spuščanja (angl. *blind drop*) [3, 4, 13].

Anonimnost na internetu lahko razdelimo na tri skupine: anonimnost pošiljatelja, anonimnost prejemnika ter anonimnost povezave.

2.1 Skupine anonimnosti

2.1.1 Anonimnost pošiljatelja

Anonimnost pošiljatelja lahko definiramo kot nezmožnost povezave sporočila z njegovim pošiljateljem. Ko se paket prenaša po omrežju, ima določen izvor (pošiljatelja) in cilj (prejemnika). Z anonimnostjo pošiljatelja se prepreči morebitnemu napadalcu identificiranje izvora, torej pošiljatelja. Anonimnost pošiljatelja ima tudi slabosti. Problem je prikriti izvorni naslov, ki vpliva na dvosmerno komunikacijo. Drugi problem oziroma slabost pa je napad lažnega predstavljanja (impersonacija) [4, 9].

- **Dvosmerna komunikacija.** Pri dvosmerni komunikaciji poznamo dve vrsti omrežnih povezav, nepovezavne (angl. *connection-less*) ter povezavne (angl. *connection-oriented*). Nepovezavna omrežja so tista, pri katerih pošiljatelj po pošiljanju ne pričakuje odgovora ali potrditve. Ta vrsta je idealna za anonimnost pošiljatelja, saj prejemnik ne potrebuje informacije o pošiljatelju. Pošiljatelj v takih omrežjih ponavadi uporabi lažni naslov. Primera protokolov nepovezavnih omrežij sta ICMP in UDP. Pri povezavnih omrežjih pa poteka dvosmerna komunikacija, torej pošiljatelj pričakuje odgovor ali potrditev prejemnika. Pošiljatelj v takem omrežju ne more uporabiti lažnega naslova za zagotovitev svoje anonimnosti, ker tako ne bo dobil pričakovanega odgovora ali potrditve. Najpogosteje se za anonimnost pošiljatelja v povezavnih omrežjih uporablja distribucijski sistem. To je sistem avtonomnih računalnikov ali programov, ki so med seboj povezani v omrežju. Glavna naloga distribucijskega sistema je reševanje zadanega problema, torej anonimnost pošiljatelja v našem primeru. Računalniki oziroma programi med seboj komunicirajo, da skupaj rešijo zadani problem.

- **Impersonacija.** Obstaja tudi možnost impersonacije drugega sistema. To pomeni, če ima pošiljatelj možnost nastaviti omrežni naslov za paket na katerokoli vrednost, lahko napadalec to izkoristi in nastavi omrežni naslov nekoga drugega in s tem zabriše sledi. Impersonacije torej omogočajo napadalcem namerno zakrivanje sledi, poleg omogočanja anonimnosti pošiljatelja. Sledenje neresničnih naslovov je dosti težje kot sledenje neveljavnih naslovov. Le-te takoj prepoznamo kot neveljavne, medtem ko neresnične ne. Impersonacija je osnova napadov DoS (*Denial of Service*). Proti napadom z uporabo impersonacije lahko uporabimo izstopni filter kot preventivni ukrep. Izstopno filtriranje je najučinkovitejše, če je locirano blizu pošiljatelja, sicer je filtriranje možno zaobiti. Z uporabo lažnega naslova lahko pošiljatelj oziroma napadalec zaobide vse izstopne filtre na drugi strani najbližjih usmerjevalnikov [4].

2.1.2 Anonimnost prejemnika

Anonimnost prejemnika je nezmožnost povezave sporočila z njegovim prejemnikom. Za razliko od anonimnosti pošiljatelja, kjer skrivanje pošiljateljevega naslova še vedno omogoča dostavo paketa, predstavlja skrivanje prejemnikovega naslova veliko težji problem. Brez določenega prejemnikovega naslova, paketa ne moremo usmeriti in dostaviti. Za anonimnost prejemnika poznamo tri glavne pristope in sicer *broadcast*, prestrezanje in višjeplastno usmerjanje.

- **Broadcast.** Broadcast in multicast omogočata večim prejemnikom sprejem istega sporočila. Morebitni napadalec lahko zoži seznam prejemnikov, ki so prejeli sporočilo, vendar ne more določiti točnega prejemnika.

- **Prestrezanje.** Pošiljatelj pošlje paket v omrežje z namenom, da ga bo prejemnik prestregel. Če prejemnikov usmerjevalnik deluje v promiskuitetnem načinu (to je način, pri katerem usmerjevalnik prestreže vse

pakete in ne samo tistih, ki so njemu namenjeni oziroma pakete broadcast/multicast) in je na poti, po kateri potuje paket, ga bo sprejel. Ta način anonimnosti prejemnika je izredno težko identificirati, kar je prednost. Vendar ima več slabosti kot prednosti. Namreč, prestrezanje je težko nastaviti (konfigurirati) in ni nujno zanesljivo. Malo omrežij vsebuje več usmerjevalnikov. Lahko pa en usmerjevalnik povezuje več omrežij, ampak ponavadi ne preusmerja prometa do naslednjega usmerjevalnika. Posledično lahko le malo gostiteljev prestreže potujoči paket. Internet je mišljen kot dinamično okolje in nihče ne more zagotoviti, da bo paket potoval po določeni poti. Problem je tudi uravnoteževanje obremenitve povezav. Namreč, če je pot, po kateri bi želeli, da paket potuje, preobremenjena, bo paket šel po drugi poti in tako onemogočil prejemnikovem usmerjevalniku prestrezanje.

- **Višjeplastno usmerjanje.** To je eden najpogostejših načinov skrivanja prejemnikovega naslova. Uporablja višje plasti modela ISO/OSI za usmerjanje podatkov. Vsak usmerjevalnik spremeni pošiljateljev in prejemnikov naslov, resnični naslov pa je zakriptiran skupaj s podatki. Ko zadnji usmerjevalnik prejme paket, dekriptira podatke in pravi naslov. Posredovalniški strežniki (angl. *proxy servers*) in čebulasti usmerjevalniki (angl. *onion routers*) so primeri višjeplastnega usmerjanja [4].

2.1.3 Anonimnost povezave

Anonimnost povezave je nezmožnost povezave pošiljatelja s prejemnikom ali povezave prejemnika s pošiljateljem. Tudi če sta pošiljatelj in prejemnik neznana, sta sledljiva zaradi korelacij dostopa. S spremljanjem usmerjevalnika lahko napadalec zoža pošiljateljevo in prejemnikovo podomrežje. Dovoli sledenje poti anonimnega paketa, vendar ne more identificirati dejanskega pošiljatelja. Vseeno lahko identificira specifične uporabnike s sledenjem večih usmerjevalnikov. Ponavadi se ta pristop uporablja za sledenje dolgoročnih

anonimnih prenosov. Premaga se ga lahko z uporabo kratkotrajnih prenosov in nenehnim spreminjanjem omrežij.

Napadalci v sredini imajo vpogled v vzpostavljene povezave, prekinjene povezave in v količino prometa. To uporabijo za statistiko uporabe kanala in korelacijo običajnih vzorcev. Na primer, imamo serijo anonimnih paketov, ki se pojavljajo vsakodnevno v osemurnih periodah. S tem lahko napadalec določi časovni pas, saj imajo ljudje večinoma osemurne delavnike od jutra do popoldneva. Če napadalec opazi veliko količino prometa, to v večini primerov pomeni, da gre za nek avtomatiziran proces ali grafično zahtevno aplikacijo. Mala količina prometa pa kaže na ročno vnašanje [4].

2.2 Anonimnost omrežja

Anonimnost omrežja je nezmožnost identifikacije pošiljatelja in prejemnika. Poznamo veliko načinov za doseganje anonimnosti omrežja. Najpogostejši so spremenljivi naslovi, sistemi slepega spuščanja, posredovalniški strežniki, usmerjanje mesh in mrežno (angl. *grid*) usmerjanje, čebulasto usmerjanje (angl. *onion routing*) in generatorji umetnega prometa (angl. *chaffing*). Spremenljivi naslovi in sistemi slepega spuščanja niso nujno priročni ali hitri. Posredovalniški strežniki in specializirana usmerjevalna omrežja (usmerjanje mesh, mrežno (grid) usmerjanje, čebulasto usmerjanje) lahko zagotavljajo višjeplastno usmerjanje. Za prikrivanje so pogosto uporabljeni generatorji umetnega prometa.

2.2.1 Spremenljivi naslovi

Spremenljivi naslovi so ena najpreprostejših oblik anonimnosti pošiljatelja in uporabljajo mobilno naslavljanje. Pred pošiljanjem najprej spremenimo omrežni naslov na drugačen, začasni, naslov, ki je na istem podomrežju in še prost. To uporabimo za anonimno povezavo. Po koncu pošiljanja (po prekinitvi povezave) ponastavimo pravi omrežni naslov. Omrežni dnevniki zaradi spremembe omrežnega naslova zabeležijo nov strežnik. Dnevniki lokal-

nega omrežja ali stikala (angl. *switch*) lahko ugotovijo, ali se je pojavil nov sistem na podomrežju ali se je le spremenil omrežni naslov. Če spremenimo še strojni naslov (angl. *MAC address*), dobimo popolnejše spreminjanje naslovov in s tem se tudi izognemo povezavi strojnega naslova z dvema omrežnima naslovoma.

Anonimni uporabnik lahko tudi razmisli o spremembi podomrežja. Preprost način za dostop do različnih omrežnih naslovov in podomrežij zagotavljajo odprti omrežni priključki, zastonj, javna ali odprta brezžična omrežja [4].

2.2.2 Sistemi slepega spuščanja

Sistem slepega spuščanja (angl. *blind drop*) je vmesni sistem, dostopen iz strani pošiljatelja in iz strani prejemnika ter začasno shrani informacije za anonimnega pošiljatelja in prejemnika. Anonimni pošiljatelj shrani sporočila v sistem slepega spuščanja med prenosom podatkov. Prejemnik se nato poveže s sistemom slepega spuščanja in prenese sporočilo. Na ta način zagotovimo anonimnost pošiljatelja in prejemnika.

Povečini sisteme slepega spuščanja uporabljajo spletni zločinci in vladni agenti. Na primer, spletni ribiči (angl. *phishers*; to so spletni zločinci, ki uporabljajo impersonacijo ponavadi za pridobivanje podatkov kreditnih kartic) uporabljajo zastonj e-poštne naslove kot sistem slepega spuščanja. Ko spletni ribič zlorabi podatke, si jih pošlje na ta e-poštni naslov. Kasneje jih anonimno prenese na svoj računalnik. S tem se zavaruje pred identifikacijo. Podobno delujejo agenti. Na primer, agent pod krinko želi sporočiti svoji agenciji pridobljene podatke. To zapiše v datoteko in jo shrani na spletno stran, ki deluje kot sistem slepega spuščanja. Kasneje vladna agencija anonimno prenese datoteko in pridobi pomembne podatke.

Vendar imajo sistemi slepega spuščanja tudi dve pomembni slabosti:

- **Hitrost.** V sistemu lahko pride do dolgih zamud z nalaganjem in prenašanjem podatkov (razen v primeru natančne sinhronizacije pošiljatelja in prejemnika). Sistemi slepega spuščanja so najboljši v časovno nekritičnih situacijah.

- **Identifikacija sistema.** Opazovalec lahko hitro prepozna sistem slepega spuščanja. Če ga nekaj časa opazuje, lahko določi tudi vzorce nalaganja in prenašanja. Temu se lahko izognemo z uporabo kriptografije.

Sistem slepega spuščanja je preprost za implementacijo in omogoča tako anonimnost pošiljatelja kot tudi anonimnost prejemnika. Vsekakor ti sistemi niso namenjeni za uporabo hitre komunikacije. Sporočila se lahko prestreže in onemogoči. S tem se prepreči dostava sporočila [4].

2.2.3 Posredovalniški strežniki

Posredovalniški strežniki delujejo kot usmerjevalni protokoli, tj. posredujejo sporočila od pošiljatelja do prejemnika. Zagotavljajo anonimnost pošiljatelja. Le-ta se poveže na posredovalniški strežnik in pri tem izpostavi le strežniški omrežni naslov. Opazovalec, postavljen med posredovalniškim strežnikom in prejemnikom lahko vidi povezavo, ampak ne more identificirati pošiljatelja. Opazovalec, postavljen med posredovalniškim strežnikom in pošiljateljem prav tako vidi povezavo, vendar ne more identificirati prejemnika.

Posredovalniški strežniki imajo več načinov uporabe, vendar sta najpogostejši dve. Prva je za pohitritev omrežnega prometa s shranjevanjem v medpomnilnik (angl. *cache*). S tem lahko posredovalniški strežnik hitreje dostavlja zahteve, saj ima večinoma strani shranjene v medpomnilniku in dostopa do zunanjih strežnikov le, kadar je res potrebno. S tem razbremeni zunanje strežnike in pohitri promet. Drugi način uporabe pa je skrivanje omrežnega naslova, torej anonimnost. Tu prideta na vrsto anonimna posredovalniška strežnika SOCKS4 in SOCKS5. Posredovalniški strežniki imajo naslove v obliki `[omrežni_naslov_strežnika]:[vrata]`.

Ponavadi so specifični glede na protokol. Najpogostejši so osnovani na protokolih SOCKS, HTTP, FTP in telnet. Pogosto se uporabljajo tudi kot izstopni filtri znotraj podjetij. Podjetja s svojimi posredovalniškimi strežniki in z enim izstopnim usmerjevalnikom v internet ustvarijo eno točko za nadzor vdorov. Poleg ostalih funkcij lahko posredovalniški strežniki filtrirajo

povezave glede na podomrežja in omrežne naslove. To pomeni, da se zaposleni v podjetju lahko povežejo na internet, napadalci, ki so na internetu pa se ne morejo povezati na posredovalniški strežnik in se tako usmeriti v podjetje.

Primerni so za hitro komunikacijo (povezovanje v realnem času), imajo pa druge omejitve. Večina posredovalniških strežnikov vzdržuje dnevnike povezav, kar pomeni, da lahko lastnik strežnika identificira pošiljatelja, prejemnika in kakršnokoli nekriptirano vsebino. Problem so tudi notranji opazovalci. Velika večina posredovalniških strežnikov ne zagotavlja enkripcije. Čeprav je vsebina med pošiljateljem in prejemnikom kodirana, mora posredovalniški strežnik biti sposoben identificirati prejemnikov naslov. Opazovalec, postavljen med posredovalniškim strežnikom in pošiljateljem bo verjetno lahko identificiral prejemnika in vsebino. Težava posredovalniških strežnikov je tudi, da se občasno spletne strani počasneje odpirajo zaradi preusmerjanja informacij. Lahko se zgodi, da se kakšni elementi spletne strani ne prikažejo, saj se posredovalniški strežnik poskuša izogniti sumljivi programski opremi. Problem, ki ni neposredna omejitev posredovalniškega strežnika, je, da lahko pridemo do uporabnikovega omrežnega naslova na drugi strani anonimnega posredovalniškega strežnika, če le-ta uporablja dodatke za brskalnik (na primer Java VM in Flash).

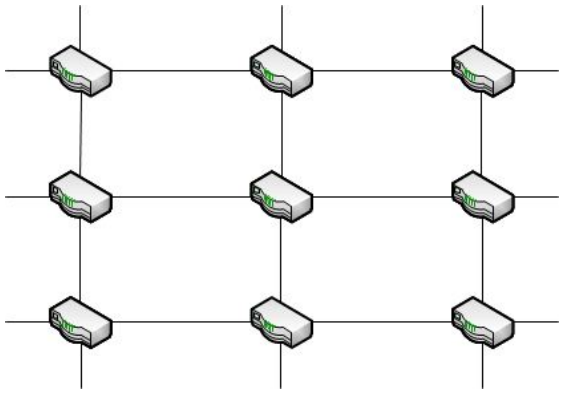
Za reševanje nekaj od teh težav lahko pošiljatelji zamenjajo več posredovalniških strežnikov pri pošiljanju. Tako veriženje omogoča anonimnost pošiljatelja, saj prepreči vmesnim posredovalniškim strežnikom beleženje pošiljateljevega omrežnega naslova. Vendar je veriga posredovalniških strežnikov statična in ima pomemben vpliv na hitrost, saj se vsak paket usmerja skozi posredovalniške strežnike, ki niso na optimalni poti do prejemnika. Če potrebuje en paket nekaj mikrosekund za potovanje od pošiljatelja do prejemnika, potem bo potreboval nekaj sekund za potovanje od pošiljatelja do prejemnika preko veriženih posredovalniških strežnikov. Še več, napadalec lahko pri kateremkoli posredovalniškemu strežniku v verigi opazuje vsebino paketa ali zaporedje povezav in s tem identificira tako pošiljatelja kot pre-

jemnika [4, 14, 21].

2.2.4 Usmerjanje mesh in mrežno (grid) usmerjanje

Omrežja mesh in mrežna (grid) omrežja so alternativa verižnim posredovalniškim strežnikom. Obe vrsti omrežja sta zasnovani za visoko razpoložljivo povezovanje. Sestavljeni sta iz večih usmerjevalnikov, ki so med seboj povezani v vnaprej določenem vzorcu. Prav v tem vzorcu se razlikujeta.

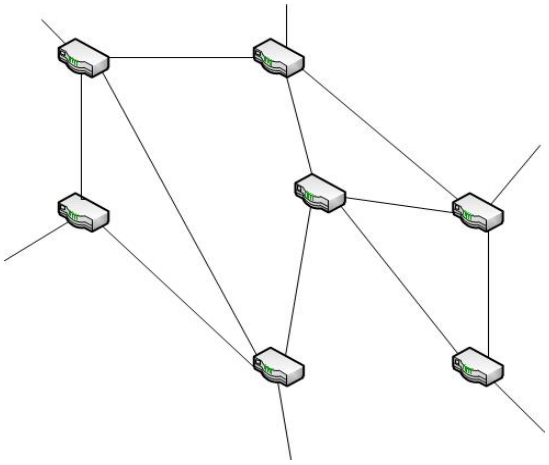
Mrežno omrežje ima usmerjevalnike (lahko so tudi računalniki) razporejene v obliki mreže, kjer ima vsak usmerjevalnik štiri povezave do sosednjih usmerjevalnikov ali priklopljenih računalnikov. Seveda je ta razporeditev kot v dvodimenzionalnem prostoru. Te mreže so lahko tudi tridimenzionalne ali n-dimenzionalne, v katerih so potem usmerjevalniki razporejeni v trikotnih, kvadratnih ali celo osmerokotnih vzorcih. V vsaki topologiji imajo usmerjevalniki isto število izhodnih povezav, ne glede na dimenzijo ali obliko vzorca. Slika 2.1 prikazuje dvodimenzionalno shemo mrežnega (grid) omrežja z devetimi usmerjevalniki.



Slika 2.1: Shema mrežnega (grid) omrežja

Za razliko od mrežnega (grid) omrežja, omrežje mesh ni tako strukturirano. Pri tem tipu omrežja imajo usmerjevalniki različno število izhodnih povezav, vendar mora biti vsak usmerjevalnik povezan na vsaj dva usmerjevalnika, ne nujno sosednja. Razlog je v uravnoteževanju obremenitve

povezav. Če je ena pot do usmerjevalnika preobremenjena, potem se paketi preusmerijo na alternativno pot. Pri omrežjih mesh poznamo dva tipa omrežij in sicer polno povezano omrežje mesh in delno povezano omrežje mesh. Slika 2.2 prikazuje primer delno povezanega omrežja mesh s sedmimi usmerjevalniki. Pri polno povezanem omrežju mesh je vsak usmerjevalnik povezan z vsakim. Torej, če imamo omrežje s petimi usmerjevalniki, so vsi med seboj povezani in vsak usmerjevalnik ima štiri izhodne povezave. Delno povezano omrežje mesh pa nima te omejitve, vendar mora imeti vsaj dve izhodni povezavi.



Slika 2.2: Shema delno povezanega omrežja mesh

Dobra stran mrežnih (grid) omrežij in omrežij mesh je ta, da če je katera povezava preobremenjena ali ni na voljo, se promet usmeri po drugi poti in tako ne prihaja do izgub. Iz vidika anonimnosti, sta ti dve omrežji idealni za zakrivanje pošiljatelja, prejemnika in vsebine povezave. Vsak paket gre od pošiljatelja do prejemnika po neznani in spreminjajoči se poti. Napadalec, lociran pri enem izmed usmerjevalnikov, lahko da sploh ne vidi prometa, lahko pa vidi le del povezave. Za večjo varnost imajo ponavadi vsi usmerjevalniki privatne ključe. Pošiljatelj v tem primeru kriptira sporočilo z javnim ključem prejemnikovega usmerjevalnika. Omrežje nato usmerja promet do prejemnika, kjer njegov usmerjevalnik sporočilo dekriptira.

Primeri anonimnega omrežja mesh so JAP (*Java Anon Proxy*), ZigBee, XO-1 in Firetide [4, 15, 16].

2.2.5 Čebulasto usmerjanje

Čebulasto usmerjanje, za razliko od usmerjanja mesh in mrežnega (grid) usmerjanja, prepusti izbiro poti pošiljatelju. Osnovno čebulasto usmerjanje poteka v štirih korakih:

- **Izbira poti.** Pošiljatelj prejme seznam usmerjevalnikov in njihove ključe za kriptiranje. Nato izbere nekaj usmerjevalnikov izmed ponujenih, preko katerih bo pošiljal podatke. Pot, ki jo izbere, se imenuje veriga.

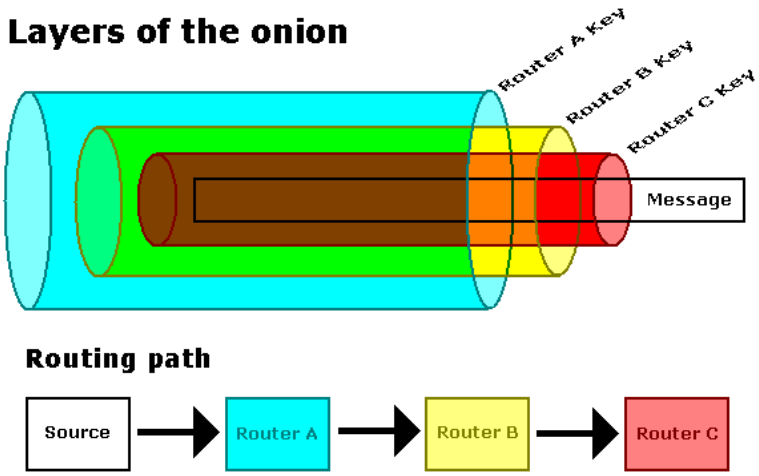
- **Kriptiranje.** Z uporabo kriptografije z asimetričnimi ključi, kriptira sporočilo z vsemi ključi izbranih usmerjevalnikov. Najprej kriptira sporočilo in ključ do predzadnjega usmerjevalnika (za morebitno dvosmerno komunikacijo) s ključem zadnjega usmerjevalnika. To kriptira skupaj s ključem predhodnega usmerjevalnika v verigi. Postopek izvede tolikokrat, kolikor usmerjevalnikov si je izbral. Tako je sporočilo dobro kriptirano z večimi plastmi. Zaradi teh plasti se to sporočilo imenuje tudi čebula (angl. *onion*). Ko je postopek kriptiranja končan, pošiljatelj pošlje čebulo po točno določeni poti.

- **Dekriptiranje.** Ko se kriptirano sporočilo prenaša, se postopoma dekriptira. Vsak usmerjevalnik dekriptira "vrhnjo plast"čebule s svojim privatnim ključem. S tem pride do naslednje plasti in izve, kateremu usmerjevalniku jo mora posredovati. Tako delno dekriptirano čebulo pošlje do naslednjega usmerjevalnika.

- **Dostava podatkov.** Zadnji usmerjevalnik v verigi dekriptira zadnjo plast in pošlje sporočilo prejemniku.

Na tak način se vsak usmerjevalnik v verigi zaveda le predhodnega usmerjevalnika (od kje je dobil sporočilo) in naslednjega usmerjevalnika (kam mora poslati sporočilo).

Na sliki 2.3 je prikazan potek dekriptiranja in pot sporočila. Pošiljatelj (na sliki *Source*) kriptira sporočilo s ključem usmerjevalnika C, to kriptira s ključem usmerjevalnika B in vse skupaj kriptira še s ključem usmerjevalnika A. Kriptirano sporočilo nato pošlje usmerjevalniku A. Le-ta dekriptira s svojim ključem in tako odstrani prvo plast čebule. Sporočilo pošlje usmerjevalniku B, ki dekriptira svojo plast in pošlje usmerjevalniku C. Usmerjevalnik C nato dekriptira zadnjo plast in vidi, komu je sporočilo namenjeno ter ga prejemniku tudi posreduje.



Slika 2.3: Shema dekriptiranja sporočila in potek poti (Vir: (2011) Wikipedia. Dostopno na: en.wikipedia.org/wiki/File:Onion_diagram.png)

Ko prejemnik prejme paket, povezava lahko ostane odprta (kar ni priporočljivo), lahko pa se zapre in se znova vzpostavi s čebulnim odgovorom (angl. *reply onion*; to je odgovor, ki ga prejemnik pošlje pošiljatelju prvega sporočila). Čebulni odgovor namesto podatkov vsebuje ključe vseh usmerjevalnikov, ki so bili na poti pri pošiljanju prve čebule. Tako pošiljatelj dobi potrditev, da je njegovo prvo sporočilo prispelo in s čebulnim odgovorom je

omogočena dvosmerna komunikacija. Čebulni odgovor se kriptira isto kot navadno sporočilo, le da namesto podatkov vsebuje ključe usmerjevalnikov. Čebulni odgovori se lahko uporabljajo tudi za omogočanje anonimnih odgovorov pošiljatelju čebulnega odgovora. Pošiljatelj pošlje čebulni odgovor in kdorkoli ga lahko prevzame in ga pošlje nazaj pošiljatelju. Tako se vzpostavi anonimna povezava med pošiljateljem in tistim, ki mu odgovarja.

Primer čebulastega usmerjevalnega sistema je Tor (*The Second-Generation Onion Router*). Narejen je bil za zagotavljanje dvosmerne povezave in za zagotavljanje anonimnosti pred analizo omrežja. To je omogočeno le s podporo povezavnih protokolov, kot je TCP. Vsaka povezava TCP vzpostavi pot skozi omrežje s posredovalniškimi strežniki in se vzdržuje, dokler se povezava ne prekine. Tor oziroma čebulasto usmerjanje zagotavlja anonimnost pošiljatelja, prejemnika in povezave.

- **Anonimnost pošiljatelja.** Pošiljatelja ne moremo identificirati iz poljubnega usmerjevalnika v čebulastem omrežju in prav tako ne iz končnega usmerjevalnika.

- **Anonimnost prejemnika.** Do zadnjega usmerjevalnika je prejemnik neznan. Zadnji usmerjevalnik je tudi edini (poleg pošiljatelja), ki vidi vsebino paketa.

- **Anonimnost povezave.** Pošiljatelj naključno izbere pot za vsako povezavo TCP posebej, zato lahko vsak čebulni usmerjevalnik spremlja le obseg posamezne povezave TCP. Za identifikacijo dolgoročnih vzorcev ima premalo informacij.

Poleg zagotavljanja anonimnosti je omrežje Tor sposobno usmerjanja povezav TCP v realnem času preko omrežij, ki niso vredna zaupanja. Tor lahko poganjamo na različnih operacijskih sistemih (recimo Linux, Windows). Opravlja tudi osnovno obliko izenačevanja obremenitve omrežja (sporočila se prenašajo po naključno izbranih poteh). Zaradi večkratnega kriptiranja je prisluškovanje zelo oteženo. Za omrežje ni nujno, da zaupa vsem sodelujočim usmerjevalnikom, prav zaradi večkratnega kriptiranja.

Čebulasto usmerjanje ima nekaj slabosti:

- **Časovna analiza.** Vsa sporočila (dohodna in odhodna), ki gredo čez malo obremenjene usmerjevalnike, lahko povežemo z opazovanjem časa od prejema do ponovnega pošiljanja. To slabost lahko odpravimo s shranjevanjem večih sporočil v medpomnilnik (angl. *buffering*). Nato jih prenesemo z uporabo psevdonaključnega algoritma za časovni izbor.

- **Prestrezanje.** Usmerjevalniki občasno odpovejo ali zapustijo omrežje. Vsaka še delujoča veriga ne more biti speljana preko usmerjevalnikov, ki so omrežje zapustili ali preko usmerjevalnikov, ki so se omrežju ravno pridružili. S tem se poveča možnost uspešne analize prometa.

- **Identifikacija pošiljatelja.** Kljub kriptiranim podatkom, lahko prvi usmerjevalnik identificira pošiljateljev omrežni naslov. To izvede tako, da primerja naslov povezave z vsemi znanimi usmerjevalniki Tor. Če naslova ni med njimi, potem je to pošiljateljev omrežni naslov, sicer je nek usmerjevalnik.

- **Nadzor sistema.** Če ima napadalec nadzor nad večimi usmerjevalniki Tor, potem lahko izvede statistični model prometa. Tako identificira pošiljatelja, prejemnika in povezave.

- **Uporaba iste poti.** Omrežje Tor je počasno pri vzpostavljanju nove poti, zato ponavadi vzpostavi eno pot in po njej poteka več povezav TCP. To pohitri prenose, saj ni potrebno za vsako povezavo TCP vzpostavljati nove poti. Težava se pa pojavi, saj se poveča nevarnost analize količine prometa.

Tor omogoča posredovalniški strežnik SOCKS (omogoča vmesnik SOCKS4a za aplikacije) in anonimizira tok TCP/IP na transportni plasti [4, 10, 12, 17].

Tor ima iz pravnega vidika nekaj zadržkov glede uporabe za nelegalne namene. Na primer, uporabimo ga lahko za pridobitev cenzuriranih informacij, organiziranje političnih aktivnosti, nepooblaščeno uhajanje občutljivih

podatkov, distribucijo nelegalne pornografije, prodajo sicer kontroliranih substanc, pranje denarja, poneverbe kreditnih kartic in krajo identitete. Tor je bil do sedaj uporabljen tudi za večje kriminalne združbe, prav tako so ga uporabili v svoje namene člani znane hekerske skupine Anonymous [19].

2.2.6 Generatorji umetnega prometa (chaffing)

Chaffing je metoda, ki generira umetni promet. Če je na omrežju le en uporabnik, ki želi pošiljati, se uporabi generatorje umetnega prometa. Umetni promet, ki ga metoda generira, zakrije pravo povezavo. Poznamo štiri tipe te metode in sicer usmerjen, sekvenčni, velikostni in količinski. Vedno manj metode chaffing je potrebne v vseh tipih, saj vse več uporabnikov uporablja določen anonimni sistem.

- **Usmerjen chaffing.** Načeloma se da hitro razločiti umetni promet od pravega. Usmerjen chaffing se posveča ravno temu. Promet generira tako, da ga usmeri po različnih poteh v omrežju in se obnaša kot pravi promet. Tako napadalec težje razloči umetni in pravi promet, posledično je zagotovljen večji nivo anonimnosti.

- **Sekvenčni chaffing.** Paketi se normalno prenašajo v pravilnem zaporedju (sekvenci). Pri sekvenčni metodi chaffing se zaporedje paketov pomeša in pošlje. Napadalec lahko razbije enkripcijo, če so paketi v istem zaporedju, torej mu s to metodo delo onemogočimo. Recimo, da imamo deset paketov v neznanem zaporedju. Sekvenčni chaffing povzroči, da če želi napadalec razbiti enkripcijo, mora najprej postaviti pakete v pravilni vrstni red, posledično mora pri 10 paketih preveriti 3628800 različnih kombinacij (deset fakulteta različnih kombinacij).

- **Velikostni chaffing.** Pri velikostni metodi chaffing spreminjamo velikost bloka, saj lahko napadalec izve nekaj o podatkih samo iz velikosti blokov. Normalno so paketi iste velikosti, le zadnji paket je malce manjši. Pri tej metodi poznamo dva načina velikosti prenašanja,

normalni in naključni. Pri normalnem načinu so vsi bloki iste velikosti. Večji bloki se razrežejo na manjše, manjši bloki pa so dopolnjeni. S tem se prepreči napadalcu, da izve velikost prenosa. Pri naključnem načinu se velikosti blokov nenehno spreminjajo. Večji bloki so lahko razrezani, manjši dopolnjeni, vendar kljub vsemu ostane nekaj sekvenc iste velikosti. Pri obeh načinih napadalec ne more vedeti velikosti prenosa.

- **Količinski chaffing.** Z opazovanjem količine prometa lahko izvemo nekaj o vsebini pa čeprav je kriptirana. Na primer, če se oddaja malo nerednih paketov in je odgovor z veliko količino paketov, ki prihajajo kot rafali, lahko predvidevamo, da se oddajajo pritiski na tipke, prejemajo pa se odgovori aplikacije. Proti temu lahko uporabimo količinski chaffing. Uporabljamo ga za generiranje poljubnih podatkov. S količinsko metodo chaffing lahko zelo zavajamo morebitne napadalce. Če pošiljamo malo nerednih podatkov, lahko s to metodo prikažemo, kot da se prenaša veliko rednih podatkov. Velja tudi obratno. Če prenašamo veliko rednih podatkov, lahko količinski chaffing uporabimo za umiritev prometa, torej, da prikažemo veliko rednih podatkov, kot da se prenaša malo podatkov. Ponavadi ga uporabimo pred vzpostavljanjem in prekinjanjem povezave, saj lahko napadalec identificira prisotnost povezave in njeno trajanje [4].

2.3 Anonimnost in ISO/OSI model

Model ISO/OSI definira sedem plasti, vsaka ima svojo značilno funkcijo. Spodnje tri plasti (fizična, povezovalna in omrežna) se ukvarjajo s komunikacijo po omrežju. Četrta plast (transportna) služi za vez med spodnjimi in zgornjimi plastmi modela, njena funkcija pa je fragmentacija in defragmentacija. Zgornje tri plasti pa so namenjene uporabniški izkušnji in asistenci upravljanja aplikacij. Tabela 2.1 prikazuje glavne naloge po plasteh modela ISO/OSI.

Pri modelu ISO/OSI se moramo zavedati, da obstaja le teoretično. V re-

Plast	Ime plasti	Naloga
1	Fizična	Medij za prenos podatkov
2	Povezovalna	Določa topologijo, kontrola pretoka, odprava napak
3	Omrežna	Pravilno potovanje paketov, kvaliteta storitev
4	Transportna	Način prenosa, fragmentacija in defragmentacija
5	Sejna	Nadzor komunikacije, določa vrsto komunikacije
6	Predstavitvena	Uskladitev različnih načinov predstavitve podatkov
7	Aplikacijska	Vmesnik med uporabnikom in modelom ISO/OSI

Tabela 2.1: Prikaz glavnih nalog plasti modela ISO/OSI

alnosti se uporablja model TCP/IP, ki je precej podoben modelu ISO/OSI. Razlika je, da ima model TCP/IP pet plasti (ponekod pišejo tudi štiri), medtem ko jih ima model ISO/OSI sedem. Glede na plasti modela ISO/OSI ima TCP/IP aplikacijsko plast, ki zajema zgornje tri plasti modela ISO/OSI, transportna plast, omrežna plast, ki sta isti kot pri modelu ISO/OSI in povezovalna plast, ki zajema spodnji dve plasti modela ISO/OSI (v primeru štirih plasti) oziroma povezovalna in fizična plast, ki sta isti kot pri modelu ISO/OSI (v primeru petih plasti). Zatorej protokoli, ki funkcijsko spadajo po modelu ISO/OSI na katero izmed zgornjih treh plasti, spadajo na aplikacijsko plast po modelu TCP/IP. Analogno, protokoli, ki funkcijsko spadajo na spodnji dve plasti modela ISO/OSI, spadajo na povezovalno plast, v primeru štirih plasti modela TCP/IP.

Poglejmo še, kako je z anonimnostjo po plasteh modela ISO/OSI:

- **Fizična plast.** Fizična plast je prva plast v modelu in njena glavna naloga je prenos podatkov oziroma bitov med dvema uporabnikoma v omrežju. Varnost na tej plasti se torej zadeva le fizičnega dostopa do naprav (računalniki, usmerjevalniki, ...) ali kablov. Protokoli te plasti ne omogočajo direktno varnosti, prav tako ne omogočajo anonimnosti.

- **Povezovalna plast.** Glavne naloge povezovalne plasti so okvirjanje datagramov, sinhronizacija podatkov, določanje prejemnikov in kon-

trola fizične plasti. Na tej plasti je, kar se tiče anonimnosti, največji problem anonimnost prejemnika. Vzrok za to je ena izmed nalog povezovalne plasti - določanje prejemnikov. Povezovalna plast potrebuje omrežni naslov prejemnika, sicer ne ve, kam usmeriti pakete. To pomeni, da na povezovalni plasti anonimnost prejemnika ne more delovati. Povezovalna plast je tudi tarča impersonacije (napadalec ponaredi strojni naslov), kar je težava anonimnosti pošiljatelja (glej poglavje 2.1.1).

- **Omrežna plast.** Omrežna plast ima glavno nalogo pri usmerjanju paketov, kvaliteti storitev (angl. *quality of service* ali *QoS*) in fragmentaciji. Bistvene varnostne težave, s katerimi se omrežna plast ukvarja, so naslavljanje prejemnika, kontrola kanala in anonimnost. Ta plast zagotavlja vse tri vrste anonimnosti (glej poglavja 2.1.1, 2.1.2, 2.1.3).

- **Transportna plast.** Transportna plast povezuje spodnje omrežne plasti in zgornje aplikacijske plasti. Ukvarja se z zanesljivim in pravilnim prenosom podatkov. Določa razstavljanje daljših paketov na manjše dele ob pošiljanju (fragmentacija) in ponovno sestavljanje ob sprejemanju podatkov (defragmentacija). Pri fragmentaciji in defragmentaciji je pomemben pravilen vrstni red paketov, ker lahko prispejo v drugačnem vrstnem redu, kot so bili poslani. Odkrivanje in odpravljanje napak je tudi naloga te plasti. Sama plast ne vključuje avtentikacijskih ali enkripcijskih mehanizmov, zato večina transportnih protokolov omogoča le omejeno varnost. Glavni na tej plasti za zagotavljanje anonimnosti in varnosti so posredovalniški strežniki.

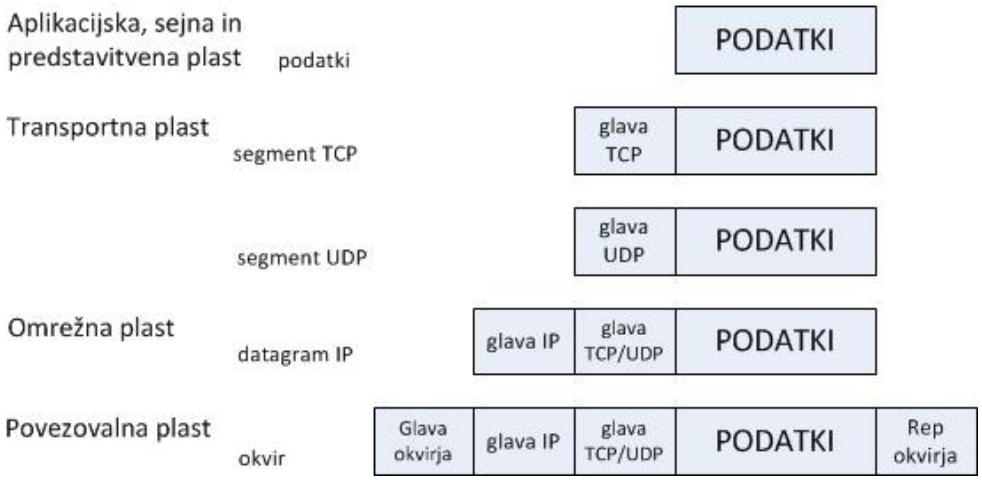
- **Sejna plast.** Sejna plast skrbi za organizacijo komunikacijskih podatkov v logične tokove. Ukvarja se tudi z dostopi in dostopnostjo nasploh. Vgrajeni ima tudi dve slabosti in sicer se pojavi problem pri avtorizaciji in dostopu. Protokoli te plasti (telnet in drugi) pošiljajo ob prijavi nezakrita uporabniška imena in gesla, kar onemogoči anonimnost pošiljatelja, prav tako se lahko vidi komu je sporočilo namen-

jeno, kar onemogoči še anonimnost prejemnika. Vidi se tudi kdo komu pošilja, kar onemogoči še anonimnost povezave. Torej sejna plast zaradi slabih ali neobstoječih avtentikacijskih mehanizmov ne zagotavlja anonimnosti. Vendar jo zagotavljajo posredovalniški strežniki na transportni plasti, kar posredno omogoča anonimnost tudi na sejni in višjih plasteh.

- **Predstavitvena plast.** Šesta (predstavitvena) plast skrbi za pretvorbo aplikacijskih podatkov v podatke, ki jih razume stroj (računalnik). Originalno je bila narejena za manipulacijo s podatki, vendar je najpogosteje uporabljena za protokole, osredotočene na varnost. Od osnovanja modela ISO/OSI sta se tehnologija in računalništvo v splošnem zelo razvila. Posledično so se pojavile potrebe po kodiranju, kompresiji in enkripciji, ki so nekakšne oblike transformacije podatkov, zato spadajo v predstavitveno plast. Anonimnost lahko na tej plasti povečini dosežemo z enkripcijo.

- **Aplikacijska plast.** Naloga aplikacijske plasti je zagotavljanje podpore omrežja protokolom, specifičnim za aplikacije. Za razliko od ostalih plasti, aplikacijska nima posebnih zahtev funkcionalnosti. Težave spodnjih plasti se prenašajo na zgornje, kar pomeni, če imamo slabo varnost in/ali anonimnost na spodnjih plasteh, se bo to zagotovo pokazalo na višjih plasteh. Velja tudi obratno. Če imamo enkripcijo oziroma varnost zagotovljeno že v aplikacijski plasti, se prenese do spodnjih plasti in tako oteži delo morebitnim napadalcem.

Na sliki 2.4 je prikazano, kako se podatki skozi plasti enkapsulirajo. Vidimo, da se omrežni naslovi prvič pojavijo šele na omrežni plasti. Za zagotavljanje anonimnosti moramo naslove prikriti. To je glavni razlog, da se anonimnost zagotavlja na omrežni plasti. Višje plasti (in transportna plast) omogočajo anonimnost le posredno, varnost pa omogočajo neposredno. Z določenimi varnostnimi prijemi zagotovimo tudi določene vrste anonimnosti [4, 11].



Slika 2.4: Shema enkapsulacije podatkov po (glavnih) plasteh modela ISO/OSI

Poglavje 3

SOCKS

SOCKS je internetni protokol, ki usmerja pakete med odjemalcem in strežnikom preko posredovalniškega strežnika. Je generičen protokol za omrežne aplikacije, osnovane na protokolu TCP/IP. Protokol SOCKS omogoča prožen okvir za razvoj varne komunikacije z enostavno integracijo ostalih varnostnih tehnologij. Samo ime je kratica za *SOCK-et-S*, torej za vtiče. Glavne značilnosti tega protokola so transparentni dostop preko večih posredovalniških strežnikov, preprosta namestitev avtentikacijskih in enkripcijskih mehanizmov, hitra namestitev novih omrežnih aplikacij in preprosto upravljanje omrežnih varnostnih politik.

SOCKS je prvič predstavil David Koblas leta 1992 na simpoziju Usenix Security Symposium. Iz prve različice sta se razvila SOCKS verzije 4 (SOCKSv4, tudi SOCKS4) in SOCKS verzije 5 (SOCKSv5, tudi SOCKS5). Glavne razlike med verzijama so predstavljene v tabeli 3.1.

Protokol	**Avtentikacija**	**TCP/UDP podpora**	**Razreševanje DNS**
SOCKS4	Ni podpore	Samo TCP	Ni vgrajeno
SOCKS5	Podpora različnih avtentikacijskih metod	TCP in UDP	Vgrajeno

Tabela 3.1: Pregled glavnih razlik med protokoloma SOCKS4 in SOCKS5

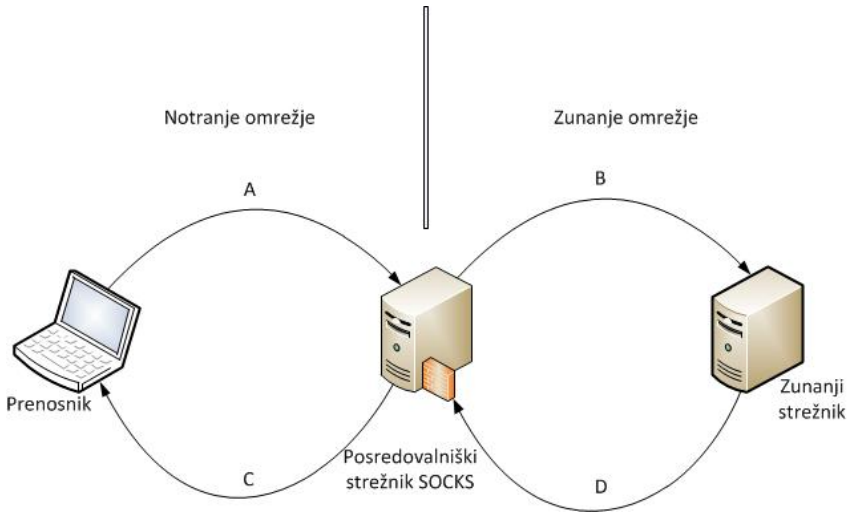
Med verzijama ni interoperabilnosti. SOCKS4 ne podpira protokola SOCKS5, SOCKS5 pa ima podporo za SOCKS4. Interoperabilnost pa se pojavi pri povezavi protokola SOCKS5 s širokim spektrom nastajajočih varnostnih modelov, kot so IPsec in PPTP/L2TP. Zaradi enostavnosti interoperabilnosti lahko rešitve, osnovane na protokolu SOCKS, hitro namestimo in uporabimo prednosti najnovejših napredkov v tehnologijah za avtentikacijo in kriptografijo, ki jih ti nastajajoči protokoli oziroma standardi ponujajo.

SOCKS4 nima uradne predstavitve v dokumentih RFC, SOCKS5 ima za razliko od SOCKS4 tri dokumente RFC. RFC 1928 opisuje protokol SOCKS, RFC 1929 opisuje avtentikacijo z uporabniškim imenom in geslom ter RFC 1961 opisuje avtentikacijo GSS-API [1, 18, 20].

3.1 Delovanje

SOCKS vključuje dve komponenti, strežnik in odjemalec. Obe komponenti sta implementirani na aplikacijski plasti. Imata pa tudi določene funkcionalnosti, glede na katere bi ju lahko uvrstili na sejno plast. Ker pa v realnosti deluje le model TCP/IP, spadata na aplikacijsko plast. Slika 3.1 prikazuje, kje je postavljen posredovalniški strežnik SOCKS in kako delujejo povezave. Ko se odjemalec želi povezati na internet, kontaktira strežnik SOCKS in z izmenjavo sporočil, definiranih v protokolu, se povezava vzpostavi. Po vzpostavitvi povezave odjemalec komunicira s strežnikom preko protokola. Po uspešni zahtevi za vzpostavitev povezave, se prične pogajanje o avtentikacijski metodi. Strežnik SOCKS pove odjemalcu, katere metode podpira. Odjemalec si nato iz seznama avtentikacijskih metod izbere tisto, s katero se želi avtenticirati. Po uspešni avtentikaciji, lahko odjemalec prične s pošiljanjem zahtev strežniku (povezava A na sliki 3.1). Ta posreduje odjemalčeve zahteve do zunanjih strežnikov (povezava B na sliki 3.1). Ti nato komunicirajo s strežnikom SOCKS, kot da je odjemalec (povezava D na sliki 3.1). Strežnik SOCKS prejete odgovore posreduje dejanskemu odjemalcu (povezava C na sliki 3.1). Zunanji strežniki se zavedajo le posre-

dovalniškega strežnika SOCKS. S tem zagotovimo anonimnost pošiljatelja. Za delovanje povezave, mora imeti odjemalec naložen del protokola, ki predstavlja odjemalca SOCKS, strežnik pa mora imeti naložen del, ki predstavlja posredovalniški strežnik SOCKS. Grobo lahko rečemo, da je SOCKS ekvivalenten vzpostavljanju tunela IP s požarnim zidom, od koder se nato sproži zahteva [1, 18, 20]



Slika 3.1: Shema lokacije posredovalniškega strežnika SOCKS in potek komuniciranja

3.2 Uporaba, prednosti in slabosti

Protokol SOCKS večinoma uporabljajo zaposleni v sektorju IT (tudi sektor informacijske tehnologije) zaradi direktnega dostopa do interneta za testiranja in odpravljanje napak. Nekatera podjetja ga uporabljajo za hitra sporočila (angl. *Instant Messaging*, IM), saj imajo skoraj vsi programi za hitra sporočila podporo protokola SOCKS. Vendar se to spreminja zaradi potreb po pregledovanju vsebine in beleženju dostopov. SOCKS namreč ne pregleduje vsebine prometa, kar je lahko slabost, če je vsebina sporna in je posredovalniški strežnik ne zazna. Zelo je uporaben tudi za inženirje in

sistemske administratorje, ker omogoča alternativno pot do interneta ali do zunanjih razmeroma varnih omrežij. Uporabimo ga tudi, če nimamo nadzora nad požarnim zidom, da bi si dovolili direkten dostop ali če nam varnostne politike ne dovolijo direktnega dostopa. S posredovalniškimi strežniki lahko tudi omejujemo brskanje po internetu. V marsikaterih podjetjih direktorji oziroma vodje ne dovolijo zaposlenim gledanja video posnetkov (npr. preko spletne strani www.youtube.com), uporabljanja socialnih omrežij (npr. Facebook, Twitter, ...) ali preprosto ne želijo, da bi zaposleni med delovnim časom brskali po ključnih besedah (npr. Wikileaks, novice, blogi, pornografija, ...). Zato uporabijo posredovalniški strežnik, s katerim lahko onemogočijo dostop do določenih spletnih strani ali onemogočijo brskanje po ključnih besedah (torej, kadarkoli bi kateri izmed zaposlenih želel brskati po ključnih besedah, posredovalniški strežnik te besede prepozna in onemogoči iskanje). V domačem okolju pa si starši dostikrat želijo imeti nadzor nad otrokovim početjem na internetu (zaradi varnosti otroka) in tu lahko uporabijo posredovalniški strežnik, s katerim vzpostavijo starševski nadzor (blokiranje določenih strani, blokiranje določenih besed, ...). Posredovalniške strežnike lahko, podobno kot čebulne usmerjevalnike, verižimo. S tem zagotovimo večjo varnost in anonimnost, saj bi se napadalec moral infiltrirati v vsak posredovalniški strežnik posebej, da bi pridobil podatke o pošiljatelju in/ali prejemniku.

Kot vsak protokol ima tudi SOCKS nekaj prednosti in nekaj slabosti.

Med prednosti štejemo anonimnost (ker se posredovalniški strežnik SOCKS "pogovarja"na internetu namesto nas, smo anonimni; glej poglavje 2), avtentikacijske mehanizme (glej poglavje 3.4), neodvisnost od aplikacij, za avtentikacijo in vzpostavljanje komunikacijske povezave potrebujemo le en komunikacijski protokol, lahko ga uporabimo bodisi za promet TCP bodisi za UDP, dodatno podpira tudi preusmerjanje prometa ICMP, ima podporo dvosmerne komunikacije in notranjega sistema NAT (angl. *Network Address Translation*, prevajanje omrežnih naslovov) za dodatno varnost in onemogočanje lažnega predstavljanja (angl. *anti-spoofing*).

Med slabosti pa štejemo nezmožnost pregledovanja vsebine (problem morebitne sporne, nelegalne vsebine), odjemalčevi programi morajo imeti zmožnost uporabe odjemalčevega dela protokola SOCKS, prav tako mora imeti zmožnost uporabe odjemalčevega dela protokola SOCKS odjemalčev operacijski sistem (prestrezanje določenih omrežnih zahtev in preusmeritev le-teh na posredovalniški strežnik SOCKS), vzdrževanje strežnika SOCKS (v preteklosti se je tu pojavila težava; vzdrževanje je velikokrat drago in časovno potratno) ipd. [1, 18, 20].

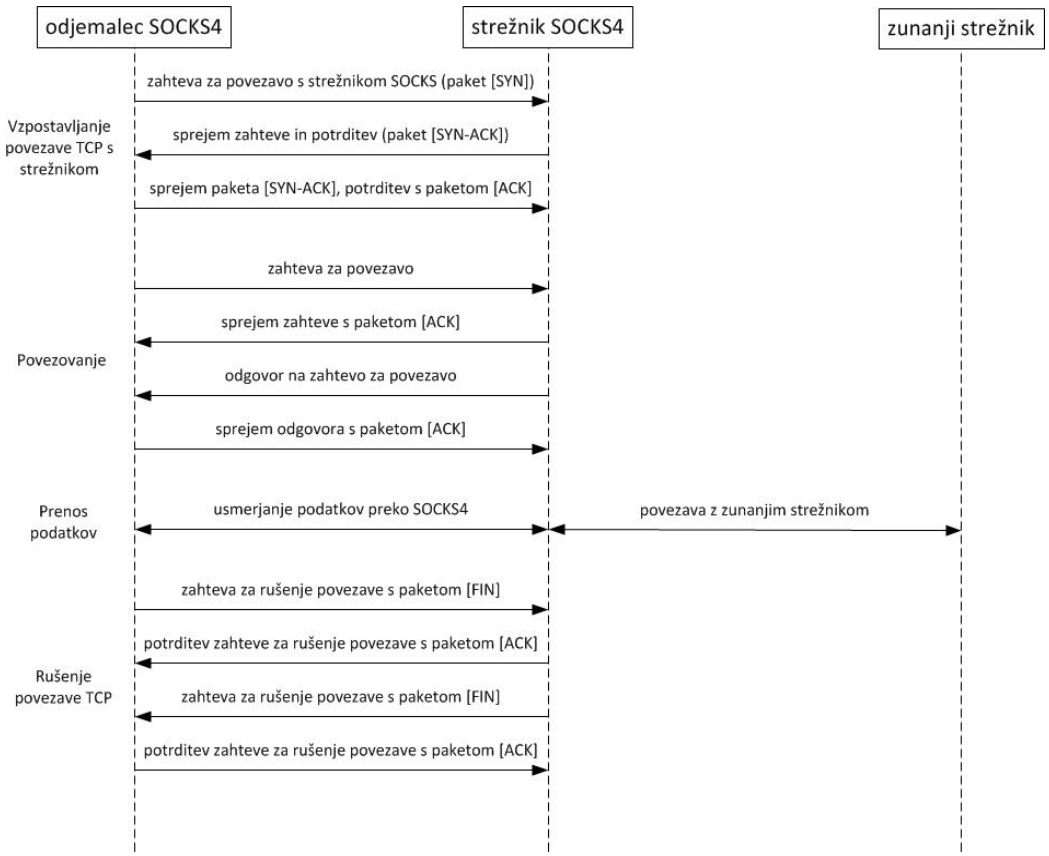
3.3 SOCKS4

SOCKS4 je, kot že rečeno, nadgradnja protokola SOCKS. SOCKS4 je protokol, ki usmerja seje TCP na požarni zid gostitelja. Tako omogoči uporabnikom aplikacij transparenten dostop preko požarnega zidu. Protokol je tudi neodvisen od aplikacij. Zaradi tega se uporablja za različne storitve, kot so na primer telnet (protokol, ki zagotavlja dvosmerno besedilno usmerjeno komunikacijo z uporabo virtualne priključitve), FTP (*File Transfer Protocol*; protokol za prenos podatkov), WHOIS (protokol, katerega glavna naloga je pridobitev kontaktnih podatkov za dodelitve omrežnih naslovov ali administratorjev domen s strani sistemskih administratorjev), www (*World Wide Web*; www je sistem medsebojno povezanih hipertekstovnih dokumentov, dostopnih preko interneta) in druge storitve. SOCKS4 ne zagotavlja varnega prečkanja požarnega zidu za aplikacije odjemalec-strežnik, osnovane na protokolu TCP, na primer zgoraj omenjeni telnet, FTP in ostali. SOCKS4 prične z nadzorom dostopa že na začetku vsake seje TCP. Strežnik nato usmerja podatke med odjemalcem in aplikacijskim strežnikom. Definiran je le za omrežne naslove tipa IPv4.

Na sliki 3.2 je prikazano rokovanje pri protokolu SOCKS4.

Pri protokolu SOCKS4 sta definirani dve operaciji, CONNECT in BIND.

CONNECT uporabi odjemalec, kadar želi vzpostaviti povezavo s strežnikom. Odjemalec uporabi BIND, ko želi vzpostaviti vhodno povezavo z



Slika 3.2: Diagram rokovanja pri protokolu SOCKS4

aplikacijskim strežnikom in po tem, ko je primarna povezava z aplikacijskim strežnikom opravljena (torej, ko se opravi CONNECT).

Tabela 3.2 prikazuje, kako je zgrajen paket CONNECT s strani odjemalca. Prva vrstica nam pove, kaj se nahaja v tistem polju in druga pove, koliko bajtov je potrebnih. Prvo polje (VER) je dolgo en bajt in je namenjeno za zapis verzije protokola (pri protokolu SOCKS4 je v tem polju številka 0x04 šestnajstiško oziroma 4 desetiško). Drugo polje (CC) je koda ukaza in nam pove ali je paket tipa CONNECT ali BIND (za CONNECT je vrednost polja CC enaka 0x01 in za BIND je enaka 0x02). Polje CC je dolgo 1 bajt. Polje DST_PORT je dolgo dva bajta in ima zapisano številko ciljnih vrat. Polje DST_IP pa v štirih bajtih definira ciljni omrežni naslov. U_ID je posebno polje, v katerem je zapisana identifikacijska številka odjemalca,

ki se želi povezati. Dolžina polja je spremenljiva, vsekakor pa mora biti zaključena z nič (torej zadnji bajt paketa mora biti 0x00 šestnajstiško oziroma 0 desetiško).

VER	CC	DST_PORT	DST_IP	U_ID
1	1	2	4	spremenljivo

Tabela 3.2: Zgradba zahteve za vzpostavitev povezave s strani odjemalca pri protokolu SOCKS4

Strežnik prejme paket CONNECT in nanj odgovori prav tako s paketom CONNECT, le da ima paket drugače definirana polja. Zgradbo takega paketa prikazuje tabela 3.3. Polje VER zopet prikazuje verzijo protokola, le da je v tem paketu nastavljeno na 0x00 oziroma 0. Polje CC prikazuje status zahteve in ima štiri možne vrednosti: 0x5a (zahteva odobrena), 0x5b (zahteva zavrnjena ali neuspela), 0x5c (zahteva neuspela, ker protokol identd ni aktiven ali ni dostopen s strani strežnika) ter 0x5d (zahteva neuspela, ker identd ni mogel potrditi identifikacijske številke v paketu), kjer je identd identifikacijski protokol, ki pomaga identificirati uporabnike določene povezave TCP. Zadnji dve polji v paketu (DST_PORT in DST_IP) sta kakršnikoli vrednosti, saj jih SOCKS4 privzeto ignorira v vseh primerih, razen v primeru odobrene zahteve (ko je vrednost polja CC enaka 0x5a). Takrat strežnik SOCKS pridobi nov vtič, ki čaka prihajajočo povezavo. Strežnik pošlje številko vrat in omrežni naslov vtiča odjemalcu v poljih DST_PORT in DST_IP. Ko odjemalec prejme odgovor strežnika, preveri polje DST_IP. Če je to polje enako 0, potem nadomesti ta omrežni naslov z omrežnim naslovom strežnika SOCKS, s katerim je povezan. Strežnik SOCKS še preveri ujemanje omrežnih naslovov v paketih CONNECT in BIND. Če ni ujemanja, prekine povezavo, sicer jo vzdržuje. Obe operaciji sta časovno omejeni. Če ne uspe povezava v določenem času, strežnik prekine povezavo.

Protokol SOCKS4 je nadgrajen v SOCKS4a. Med njima je le ena razlika in sicer protokol SOCKS4a ima možnost definiranja ciljnega domenskega imena, če ga ne more sam razrešiti. Zaradi tega se razlikujeta tudi v zgradbi

VER	CC	DST_PORT	DST_IP
1	1	2	4

Tabela 3.3: Zgradba odgovora s strani strežnika pri protokolu SOCKS4

paketov CONNECT in BIND. Tabela 3.4 prikazuje zgradbo paketa CONNECT pri protokolu SOCKS4a. Polja VER, CC, DST_PORT in U_ID so enaka kot pri protokolu SOCKS4. Polje DST_IP mora biti nujno nastavljeno na napačen omrežni naslov (tako vemo, da gre za protokol SOCKS4a in ne za SOCKS4) in sicer je oblike 0.0.0.X, kjer je X poljubna neničelna številka (v paketu ta naslov izgleda kot 0x00 0x00 0x00 0xXX, kjer je bajt 0xXX neničelen). Zadnje polje pa predstavlja domensko ime gostitelja, kamor se želimo povezati. Tako kot U_ID je spremenljive dolžine in mora biti zaključen z 0.

VER	CC	DST_PORT	DST_IP	U_ID	DN
1	1	2	4	spremenljivo	spremenljivo

Tabela 3.4: Zgradba zahteve za vzpostavitev povezave s strani odjemalca pri protokolu SOCKS4a

Tabela 3.5 prikazuje zgradbo paketa CONNECT s strani strežnika pri protokolu SOCKS4a. Polja VER, CC in DST_PORT so enaka kot pri protokolu SOCKS4. Razlika je v polju DST_IP. Strežnik, ki uporablja SOCKS4a, mora preveriti polje DST_IP in če je vrednost enaka 0.0.0.X (kjer je X neničelen), potem strežnik prebere prejeto domensko ime (poslal mu ga je odjemalec v paketu CONNECT) in se poskuša nanj povezati [5, 6, 18].

VER	CC	DST_PORT	DST_IP
1	1	2	4

Tabela 3.5: Zgradba odgovora s strani strežnika pri protokolu SOCKS4a

3.4 SOCKS5

Protokol SOCKS5 je nadgradnja protokola SOCKS4 (oziroma SOCKS4a). Ponuja različne metode za avtentikacijo, ima podporo za IPv6 in promet UDP, s katerim lahko razrešuje DNS. SOCKS5 je tudi odobren standard IETF (definiran v RFC 1928, RFC 1929 ter RFC 1961). Zagotavlja nam varnost, zanesljivost in odgovornost ter povečuje nadzor omrežja in njegovo upravljanje. Narejen je, da omogoča okvir za aplikacije odjemalec-strežnik, tako v domeni TCP kot v UDP. S tem omogoča prikladno in varno uporabo storitev omrežnega požarnega zidu, kar vključuje tudi varno prečkanje. Protokol se nahaja med transportno in aplikacijsko plastjo, natančneje, lociran je na sejni plasti. Zaradi tega ne omogoča prehodnih storitev omrežne plasti (npr. posredovanje/preusmerjanje sporočil ICMP). Implementacije protokola SOCKS5 ponavadi vključujejo rekompilacijo ali prevezovanje odjemalskih aplikacij, osnovanih na protokolu TCP, za uporabo primerne enkapsulacije v knjižnici SOCKS.

Povezovanje se začne s pozdravom (angl. *greeting*). Odjemalec pošlje strežniku pozdrav, v katerem so naštete vse avtentikacijske metode, ki jih podpira. Strežnik nato izbere eno izmed ponujenih metod ali pošlje zavrnitev, če nobena od ponujenih metod ni sprejemljiva. Nato se pošlje še nekaj paketov. Koliko, je odvisno od izbrane avtentikacijske metode. Po tem odjemalec pošlje zahtevo za povezavo podobno kot pri protokolu SOCKS4 in strežnik odgovori podobno kot pri SOCKS4. SOCKS5 vsebuje tri operacije, CONNECT, BIND in UDP ASSOCIATE. CONNECT in BIND sta podobna kot v protokolu SOCKS4.

SOCKS5 podpira naslednje avtentikacijske metode (v oklepaju je koda posamezne metode): brez avtentikacije (0x00), GSSAPI (0x01), avtentikacija z uporabniškim imenom in geslom (0x02), metode, dodeljene s strani organizacije IANA (0x03-0x7F) in metode, rezervirane za privatno uporabo (0x80-0xFE).

Pozdrav, ki ga pošlje odjemalec strežniku je prikazan v tabeli 3.6. Polje VER predstavlja verzijo protokola (v tem primeru je vrednost 0x05 šestnajst-

iško oziroma 5 desetiško) in je dolgo en bajt. Prav takšno dolžino ima tudi polje NAM, ki nam pove število podprith avtentikacijskih metod. Zadnje polje, polje AM, je spremenljive dolžine, ker so v tem polju naštete vse kode avtentikacijskih metod, ki so podprte (vsaka koda zasede en bajt).

VER	NAM	AM
1	1	spremenljivo

Tabela 3.6: Zgradba pozdrava s strani odjemalca pri protokolu SOCKS5

Strežnik na pozdrav odgovori z rahlo drugačnim paketom (glej tabelo 3.7). VER polje še vedno predstavlja verzijo protokola, ki je v tem primeru 0x05. Dolžina polja je en bajt. Drugo polje (AM) pa predstavlja kodo izbrane avtentikacijske metode in je prav tako dolgo en bajt. Če strežnik v polje AM vpiše vrednost 0xFF, pomeni, da nobena izmed ponujenih avtentikacijskih metod ni ustrezna.

VER	AM
1	1

Tabela 3.7: Zgradba pozdrava s strani strežnika pri protokolu SOCKS5

Po uspešni avtentikaciji, odjemalec pošlje zahtevo za povezavo (paket CONNECT). Zgradba tega paketa je predstavljena v tabeli 3.8. Polje VER zopet vsebuje vrednost 0x05. Polje CC predstavlja, tako kot pri SOCKS4, kodo ukaza. Ta lahko zasede vrednosti 0x01 (TCP CONNECT), 0x02 (TCP BIND) ali 0x03 (UDP). Tretje polje je rezervirano in mora imeti vrednost 0x00. Polje AT vsebuje informacijo o tipu omrežnega naslova (0x01, če gre za IPv4 naslov, 0x03, če gre za domensko ime in 0x04, če gre za IPv6 naslov) in ima dolžino enega bajta. Naslednje polje (DST_IP) pa ima določeno dolžino glede na polje AT, saj polje DST_IP vsebuje omrežni naslov. Če imamo tip omrežnega naslova IPv4, ima polje DST_IP dolžino štiri bajte, če imamo domensko ime, potem en bajt predstavlja dolžino domenskega imena in temu sledi domensko ime, v zadnjem primeru, torej če imamo IPv6, pa je dolžina

polja DST_IP enaka šestnajstim bajtom. Polje DST_PORT je, tako kot pri SOCKS4, namenjeno številki vrat v dolžini dveh bajtov.

VER	CC	0x00	AT	DST_IP	DST_PORT
1	1	1	1	spremenljivo	2

Tabela 3.8: Zgradba zahteve za vzpostavitev povezave s strani odjemalca pri protokolu SOCKS5

Strežnik na prejet paket CONNECT odgovori s svojim paketom CONNECT. Zgradba le-tega je predstavljena v tabeli 3.9. Prvo polje ima vrednost 0x05. Drugo polje (CC) predstavlja status zahteve in lahko zaseda vrednosti 0x00 (zahteva odobrena), 0x01 (splošna napaka), 0x02 (povezava ni dovoljena s strani seznama pravil), 0x03 (omrežje nedosegljivo), 0x04 (gostitelj nedosegljiv), 0x05 (povezava zavrnjena s strani ciljnega gostitelja), 0x06 (pretečen TTL - *Time To Live*), 0x07 (ukaz ni podprt ali napaka protokola) ali 0x08 (tip naslova ni podprt). Tretje polje je zopet nastavljeno na 0x00, ker je rezervirano. Polja AT, DST_IP in DST_PORT so ista kot pri paketu CONNECT na strani odjemalca.

VER	CC	0x00	AT	DST_IP	DST_PORT
1	1	1	1	spremenljivo	2

Tabela 3.9: Zgradba odgovora s strani strežnika pri protokolu SOCKS5

SOCKS5 ima, za razliko od protokola SOCKS4, še tretjo operacijo. UDP ASSOCIATE se uporablja za vzpostavitev povezave znotraj procesa preusmerjanja prometa UDP za obdelavo datagramov UDP. Ta operacija nima posebne zgradbe, razen če imamo odjemalce, osnovane na protokolu UDP. V tem primeru mora odjemalec poslati svoje datagrame UDP posredovalnemu strežniku na vrata, definirana v odgovoru UDP ASSOCIATE. Vsak datagram vsebuje glavo zahteve UDP. Ta zahteva ima svojo zgradbo (glej tabelo 3.10). Prva dva bajta sta rezervirana, zato sta nastavljena na 0x0000. Polje FRN predstavlja trenutno fragmentacijsko številko oziroma določa, ali je ta datagram eden izmed številnih fragmentov. Če je vrednost 0x00, pomeni, da

paket ni del fragmentacije. Vrednosti med 0x01 in 0x7F predstavljajo zaporedno številko fragmenta. Polja AT, DST_IP in DST_PORT so enaka kot pri zgoraj opisanih zgradbah. Polje DATA, ki je spremenljive dolžine predstavlja podatke, ki se prenašajo.

0x0000	FRN	AT	DST_IP	DST_PORT	DATA
2	1	1	spremenljivo	2	spremenljivo

Tabela 3.10: Zgradba glave zahteve UDP

Poglejmo si še zgradbi paketov za avtentikacijo z uporabniškim imenom in geslom, ki je najpogosteje uporabljena avtentikacijska metoda. Tabela 3.11 predstavlja avtentikacijsko zahtevo iz strani odjemalca, saj se s tem prične podpogajanje. Polje VER vsebuje trenutno verzijo podpogajanja, ki je 0x01, zato je tudi vrednost polja VER enaka 0x01. V polju UNL je zapisana dolžina uporabniškega imena, ki je zapisano v polju UN. Sledita še dve polji in sicer PWL, v katerem je zapisana dolžina gesla in PW, v katerem se nahaja geslo, ki je povezano z uporabniškim imenom.

VER	UNL	UN	PWL	PW
1	1	spremenljivo	1	spremenljivo

Tabela 3.11: Zgradba paketa za avtentikacijo z uporabniškim imenom in geslom s strani odjemalca

Strežnik na odjemalčev paket odgovori z zelo kratkim odgovorom (glej tabelo 3.12), kjer ima polje VER vrednost 0x01, polje CC pa ima vrednost 0x00, če je avtentikacija uspela in katerokoli drugo vrednost, če avtentikacija ni uspela in v tem primeru zapre povezavo.

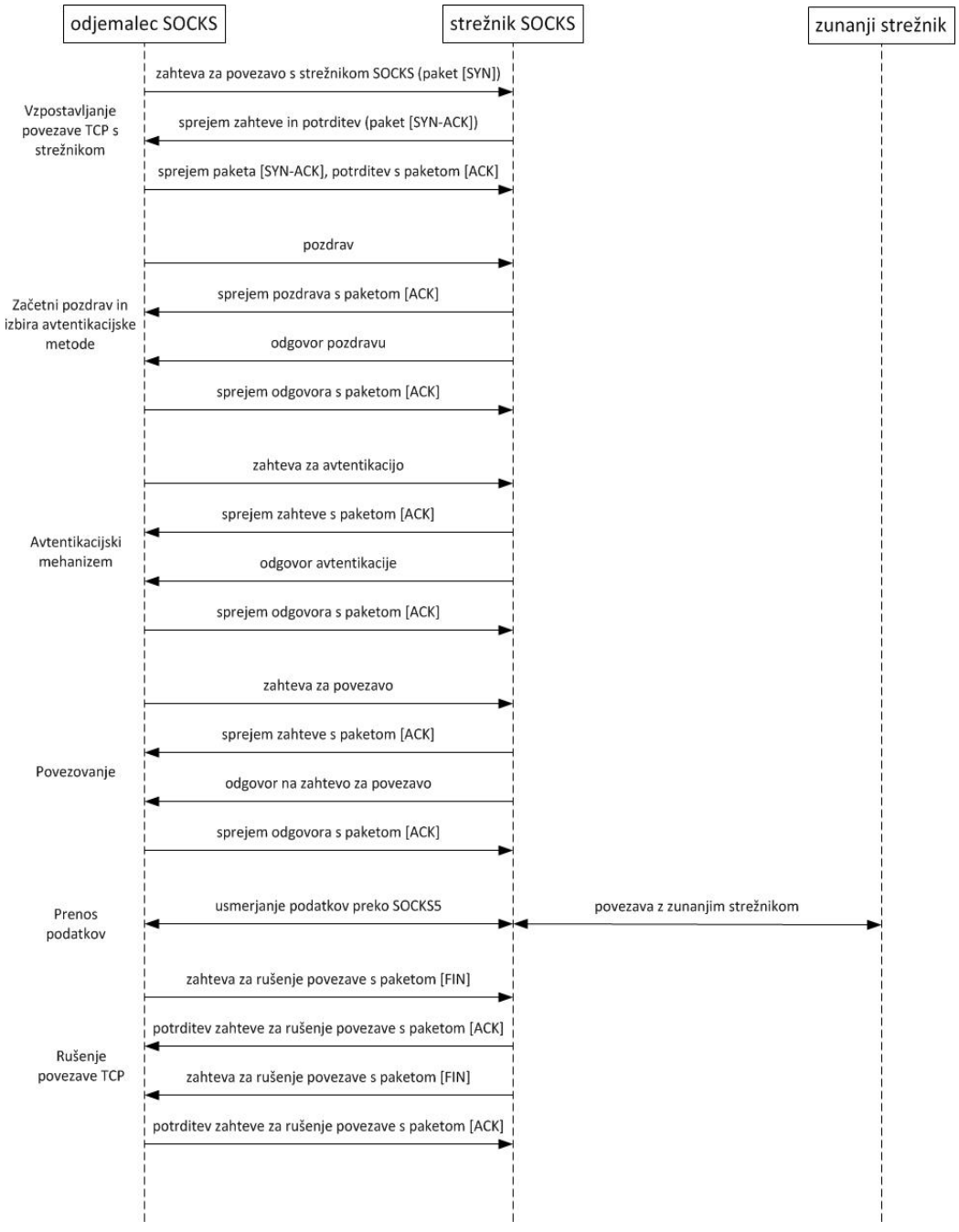
VER	CC
1	1

Tabela 3.12: Zgradba paketa za avtentikacijo z uporabniškim imenom in geslom s strani strežnika

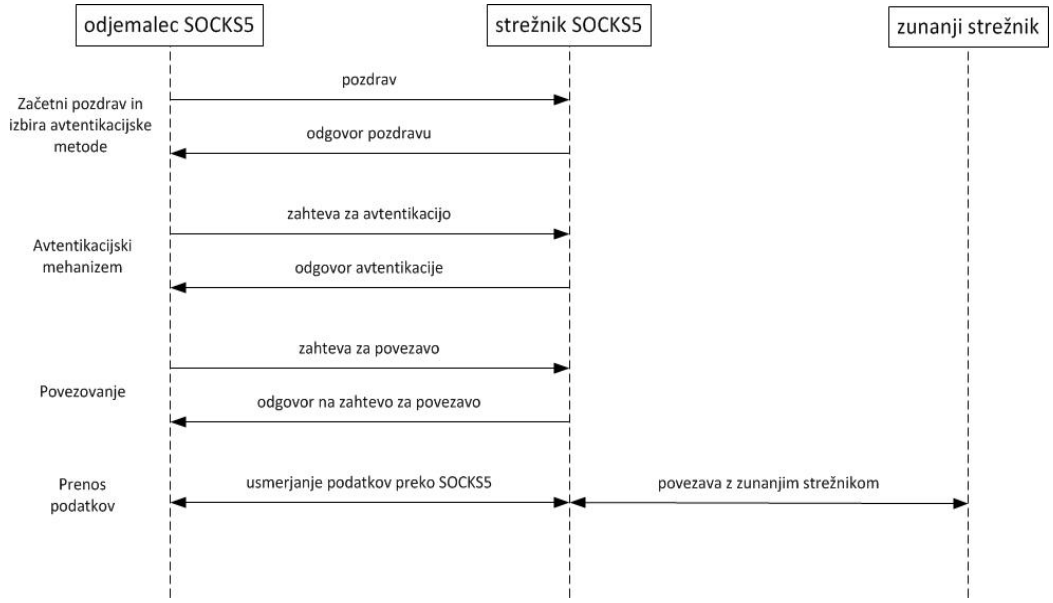
Ta način avtentikacije ni priporočljiv za sisteme, kjer obstaja možnost vohljanja (angl. *sniffing*). Razlog je v prenašanju gesla, ki se ne kriptira.

Opisan protokol je namenjen prečkanju omrežnih požarnih zidov z uporabo aplikacijske plasti. Varnost prečkanja je odvisna predvsem od avtentikacijskih in enkapsulacijskih metod, ki so omogočene v določeni implementaciji in tudi izbrane v pogajanju med odjemalcem in strežnikom. Način avtentikacije je potrebno izbrati s previdnostjo [1, 7, 8, 18].

Prikaz rokovanja za promet TCP prikazuje slika 3.3. Pregled rokovanja za promet UDP prikazuje slika 3.4.



Slika 3.3: Diagram rokovanja pri protokolu SOCKS5 za promet TCP



Slika 3.4: Diagram rokovanja pri protokolu SOCKS5 za promet UDP

Poglavje 4

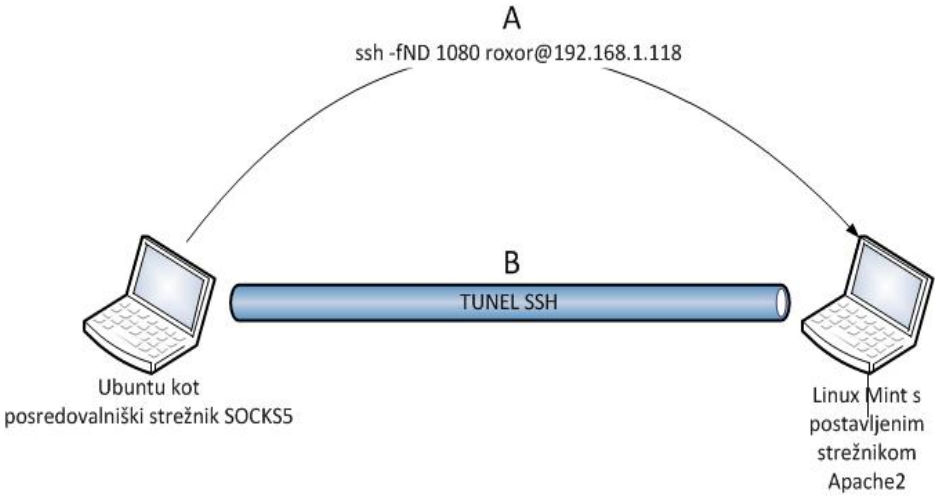
Prikaz uporabe posredovalniškega strežnika SOCKS5

4.1 Uporabljena orodja

Za prikaz uporabe smo uporabili dva prenosna računalnika. Na enem je nameščen operacijski sistem Ubuntu 11.10, na drugem pa Linux Mint 13. Za prikaz uporabe smo uporabili še strežnik OpenSSH (za postavitev posredovalniškega strežnika ter vzpostavitev tunela), strežnik Apache2 (za testno prenašanje datotek preko spletnega vmesnika) in spletni brskalnik Chromium. Po želji lahko uporabimo še prevajalnik za programski jezik Perl. Za zajemanje prometa smo uporabili program Wireshark verzije 1.6.2.

4.2 Prikaz uporabe

Za praktični del smo z uporabo ukaza SSH vzpostavili tunel med dvema računalnikoma in zajeli promet, da smo se prepričali o kriptiranju prometa. Slika 4.1 prikazuje testno situacijo, ki posnema resnično situacijo, opisano v nadaljevanju.



Slika 4.1: Shema testne situacije

Najprej smo na računalniku z operacijskim sistemom Linux Mint postavili strežnik Apache2 z omrežnim naslovom 192.168.1.118. Na levi strani imamo računalnik z operacijskim sistemom Ubuntu, ki deluje kot posredovalniški strežnik SOCKS5. Le-ta se postavi na računalniku z ukazom, prikazanim na sliki 4.2 oziroma na sliki 4.1 nad puščico v smeri desnega računalnika z operacijskim sistemom Linux Mint.

```
vesnag@SheldonBuntu:~$ ssh -fND 1080 roxor@192.168.1.118
roxor@192.168.1.118's password:
vesnag@SheldonBuntu:~$
```

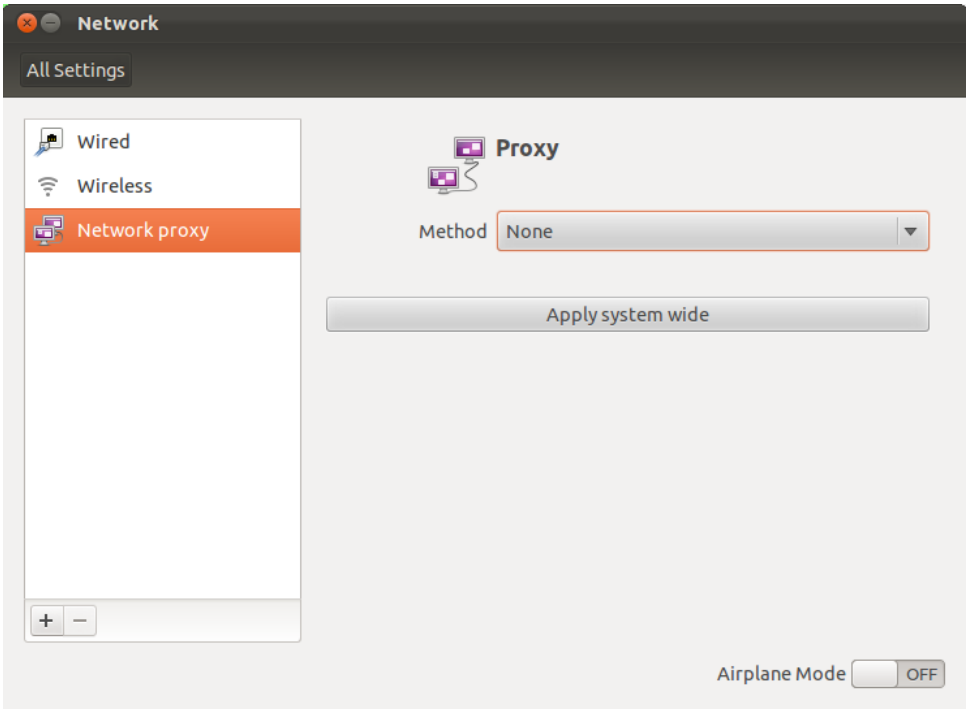
Slika 4.2: Prikaz ukaza in njegov odziv

Stikalo `-f` v ukazu pomeni, da se bo program izvajal v ozadju kot demon. Stikalo `-N` je namenjeno temu, da program ne izvede nobene akcije na domačem računalniku. `-D` pa vključi dinamično posredovanje vrat (angl. *dynamic port forwarding*). Številka 1080 naroči ukazu ssh, da postavi posredovalniški strežnik SOCKS5 na domačem računalniku, zadnji del ukaza pa ustvari tunel SSH med domačim računalnikom in drugim računalnikom ter ves promet kriptira in usmerja preko drugega računalnika. Po izvedbi ukaza

se je torej ustvari posredovalniški strežnik SOCKS5 na računalniku z operacijskim sistemom Ubuntu ter tunel med tem in računalnikom z operacijskim sistemom Linux Mint.

Na spletni vmesnik strežnika Apache2 smo postavili besedilno datoteko, ki smo jo uporabili za prenašanje. Izvedli smo dva scenarija in sicer najprej brez posredovalniškega strežnika SOCKS5 in tunela SSH ter potem še s posredovalniškim strežnikom in tunelom.

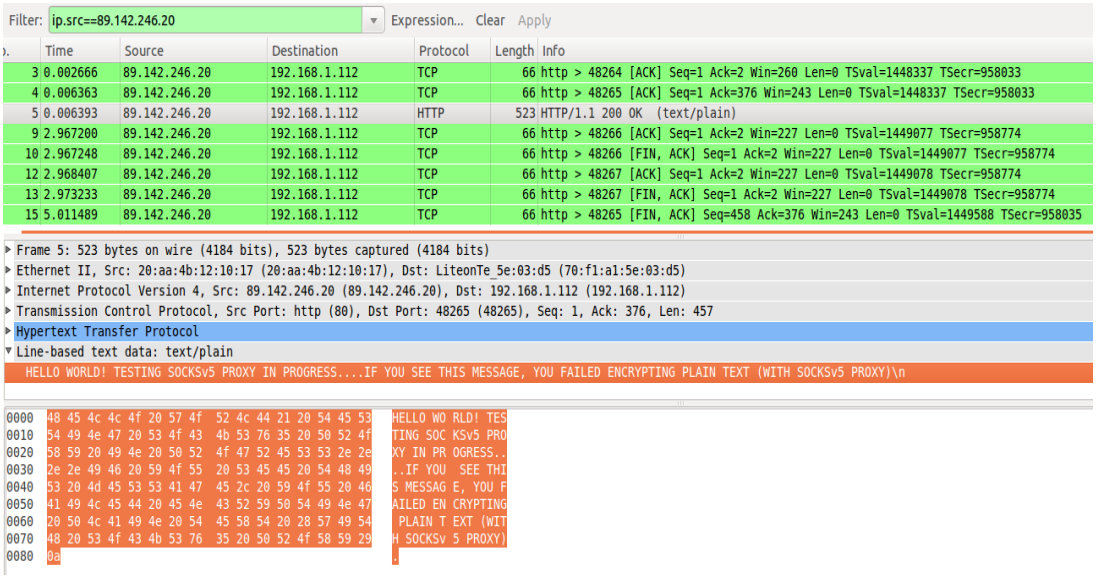
- **Prvi scenarij.** Kot že rečeno, je prvi scenarij brez posredovalniškega strežnika in tunela. V brskalniku Chromium je vidno iz nastavitev, da na njem privzeto ne deluje noben posredovalniški strežnik (glej sliko 4.3).



Slika 4.3: Nastavitve posredovalniških strežnikov brskalnika Chromium za prvi scenarij

Nato se povežemo na strežnik Apache2 (nahaja se na omrežnem naslovu `89.142.246.20`), ki deluje na drugem računalniku, preko spletnega

vmesnika na brskalniku Chromium. Zaženemo Wireshark in začnemo poslušati promet na vmesniku wlan0. Medtem, ko je zajemanje prometa v teku, prenesemo testno datoteko preko strežnika Apache2. Počakamo še kakšno sekundo in ustavimo zajemanje prometa. Slika 4.4 prikazuje zajete pakete med prenosom. Uporabili smo filter za prikaz paketov s pošiljateljevim omreženim naslovom (`ip.src==89.142.246.20`) zato, da prikažemo le naš promet. Če pogledamo sliko, vidimo, da je peti paket (osenčen na sliki) potrditev, da se je datoteka uspešno prensela. Ker nimamo dodatne varnosti, lahko tudi vidimo celotno besedilo testne datoteke, ki je na sliki v spodnji polovici označena oranžno.

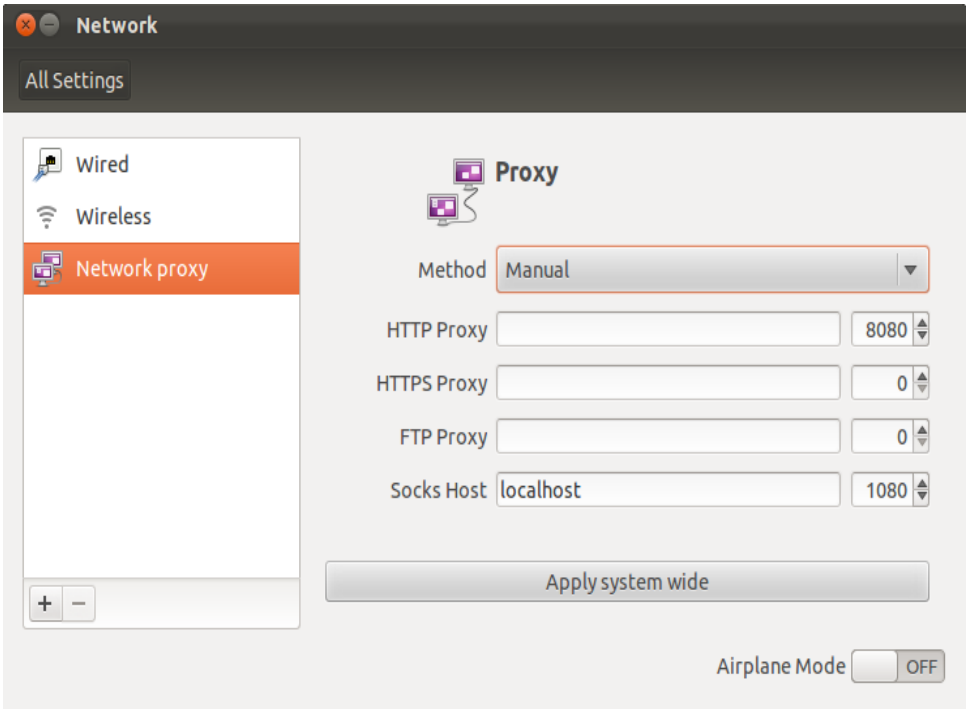


Slika 4.4: Prikaz zajema prometa s programom Wireshark za prvi scenarij

Opazimo lahko, da brez posredovalniškega strežnika in tunela hitro vidimo, kaj se po omrežju prenaša in s tem nam lahko morebitni napadalec škoduje, saj vidi, kakšen promet in kaj se prenaša po omrežju.

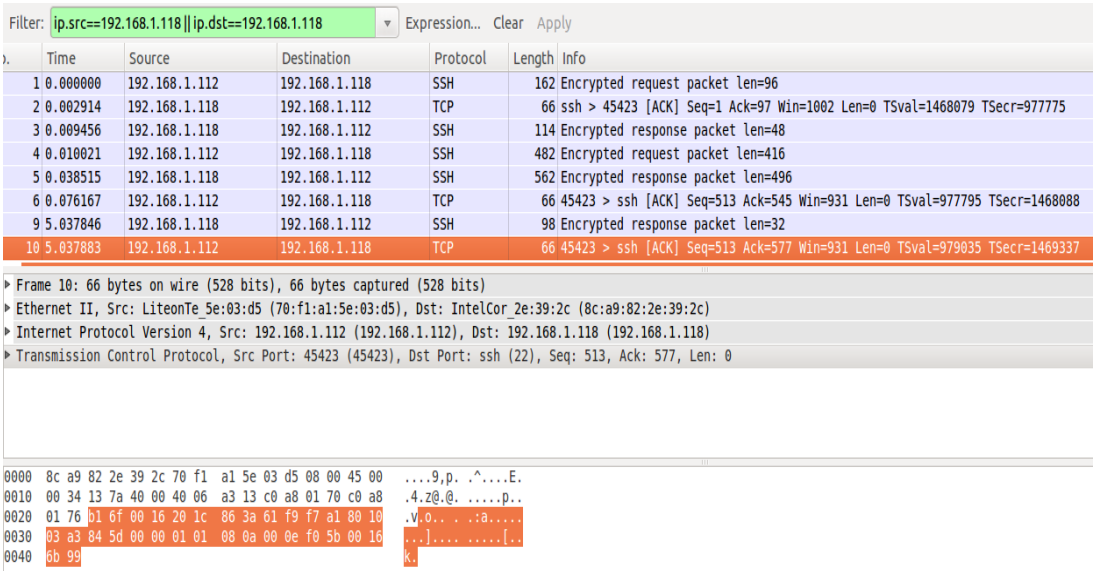
- **Drugi scenarij.** Pri drugem scenariju smo vzpostavili tunel in postavili posredovalniški strežnik SOCKS5. Če želimo pošiljati promet skozi tunel, moramo programom, ki jih uporabljamo, to povedati. To storimo

tako, da jim v nastavitvah spremenimo nastavitve posredovalniških strežnikov. Na sliki 4.5 vidimo, kako nastavimo, da bo brskalnik Chromium pošiljal promet skozi tunel. Najprej povemo, da bomo spremembe ročno vstavili (v spustnem meniju izberemo možnost *Manual*). Nato vpišemo v vrstico *Socks Host* omrežni naslov posredovalniškega strežnika SOCKS5 ali (če je posredovalniški strežnik SOCKS5 postavljen na domačem računalniku) zapišemo `localhost`. V zadnji okvirček moramo zapisati še številko vrat, na katerih deluje strežnik. Privzeta vrata za posredovalniški strežnik SOCKS5 so 1080. V našem primeru uporabimo kar privzeto vrednost. Na koncu ne smemo pozabiti shraniti sprememb. To storimo s pritiskom na gumb *Apply system wide*. S tem smo povedali brskalniku Chromium, naj deluje skozi vzpostavljen tunel.



Slika 4.5: Nastavitve posredovalniških strežnikov brskalnika Chromium za drugi scenarij

Naslednji korak pri testiranju je zajemanje prometa. Podobno kot v prvem scenariju zaženemo program Wireshark in pričnemo zajemati promet na vmesniku wlan0. Ko zajemanje poteka, ponovno prenesemo testno datoteko preko spletnega vmesnika strežnika Apache2. Ko je prenešena, počakamo še nekaj sekund in končamo z zajemanjem prometa. Rezultat našega zajemanja vidimo na sliki 4.6. Prva stvar, ki jo opazimo, je to, da v zajemu ni več prometa HTTP, ampak le še promet SSH in TCP. Razvidno je tudi, da vidimo le izmenjavo paketov med računalnikom, ki poganja posredovalniški strežnik SOCKS5 in računalnikom, ki ima postavljen strežnik Apache2. Program Wireshark ni uspel zajeti paketa s testno datoteko. Vzrok temu je tunel. Namreč preko tunela Wireshark ne zazna pravega prometa HTTP. Vsi paketi, ki so se prenesli, so kriptirani. Wireshark tudi ne zazna vedno prometa, če uporabljamo tunel.



Slika 4.6: Prikaz zajema prometa s programom Wireshark za drugi scenarij

Morebitni napadalec, kot vidimo, tudi s pomočjo Wireshark programa ne more videti prometa in tako nam ne more škodovati.

Resnični scenarij za tak primer uporabe je sledeč. Recimo, da smo

nekje na dopustu, v hotelu z možnostjo nekriptiranega brezžičnega interneta in nas kontaktirajo iz službe, da je nekaj šlo narobe ter je treba popraviti. Zaradi varovanja podatkov ne smemo kar preko nekriptiranega omrežja dostopati do službenega računalnika. Ker imamo na službenem računalniku omogočen dostop iz zunanjega omrežja, si postavimo na hotelskem ali prenosnem računalniku (če ga imamo s seboj na dopustu) posredovalniški strežnik SOCKS5 z zgoraj opisanim ukazom ter vzpostavimo tunel s službenim računalnikom. Ne smemo še pozabiti programom povedati, naj delujejo skozi tunel. Tako lahko brez skrbi dostopamo do službenega računalnika in popravimo napako. Za dodatno varnost lahko po postavitvi strežnika SOCKS5 in pred prenosom podatkov zaženemo še ovojnico TCP (angl. *TCP wrapper*; to je program, s katerim lahko, brez pravic skrbnika, nadziramo, kateri omrežni naslovi lahko dostopajo do posredovalniškega strežnika SOCKS5). Za tak scenarij bi lahko uporabili tudi tunel VPN (*Virtual Private Network*). Enkripcija je pri obeh enako močna. Razlika med tunelom SSH in tunelom VPN je v tem, da tunel VPN deluje na nivoju operacijskega sistema (to pomeni, da se ves promet preusmeri skozi tunel - operacijski sistem se obnaša kot, da bi bil na oddaljenem omrežju), medtem ko tunel SSH deluje le na nivoju aplikacij (kot že rečeno, moramo vsaki aplikaciji ali programu posebej povedati, naj deluje preko posredovalniškega strežnika SOCKS, ki deluje s pomočjo tunela SSH). Če ponovno pogledamo resnični scenarij, bi lahko postavili tunel SSH ali tunel VPN. VPN bi postavili v primeru, da v podjetju deluje strežnik VPN preko katerega se lahko povežemo in če delo od nas zahteva dostop do datotek v skupni rabi z ostalimi zaposlenimi, dostop do omrežnih tiskalnikov ali dostop do kakršnekoli druge pomembne informacije ter se ne želimo ukvarjati z nastavljanjem vsakega programa posebej. V primeru, da ne potrebujemo takega dela, lahko postavimo tunel SSH in vsak program, ki ga potrebujemo, posebej nastavimo za delovanje preko posredovalniškega strežnika SOCKS5. Iz vidika varnosti oziroma anonimnosti, sta oba enako zanesljiva. Problem pri tunelu VPN je kompliciranо postavljanje strežnika VPN. Tu je tunel SSH in delovanje preko posredovalniškega strežnika SOCKS5 veliko lažje, saj se

strežnik SOCKS5 postavi hitro. Samo delovanje preko posredovalniškega strežnika pa je lahko za začetnike zastrašujoče, za razliko od tunela VPN, kjer se začetniki brez težav povežejo na strežnik VPN (ne znajo pa ga, kot že rečeno, postaviti) [2]. Če bi imeli v našem resničnem scenariju dostop do strežnika VPN, bi lahko postavili tudi tunel VPN.

Na splošno lahko to uporabimo v katerikoli situaciji, kjer želimo iz ene lokacije dostopati do računalnika na drugi lokaciji.

Ta način nima več veliko možnosti za nadaljni razvoj, pa vendar se vedno kaj najde za nadgradnjo. V našem primeru bi lahko namesto programskega jezika Perl uporabili programski jezik Bash in tako bi lahko vse ukaze spravili v eno datoteko. S tem bi porabili malo manj časa za izvedbo ukazov in lažje bi bilo iskanje morebitnih napak. Ljudje imamo najraje programe, ki jih usposobimo za delovanje s parimi kliki in z malo razmišljanja. Pri tem načinu pa bo vedno problem, saj je potrebno vsakemu programu posebej povedati, naj deluje skozi tunel in podati mu moramo vse podatke o tunelu. Še ena izmed možnih nadgradenj pa vključuje kombiniranje opisanih mehanizmov. V našem primeru smo pokazali le kriptiranje preko posredovalniškega strežnika, lahko pa bi dodali še anonimnost pošiljatelja z uporabo spremenljivih naslovov. Ta način je primeren za računalniško izobražene ljudi. Neprimeren pa je za računalniško neizobražene, saj jim je težko razložiti, kaj morajo spremeniti, da bo delovalo. Obstaja še ena možnost nadaljnega razvoja in sicer nadgraditi protokol SOCKS5, da bi imel vključeno enkripcijo prometa.

Poglavje 5

Zaključek

V nalogi je preučenih nekaj mehanizmov, s katerimi lahko uporabnikom zagotovimo določeno stopnjo anonimnosti. Izmed mnogih varnostnih mehanizmov sem jih izbrala šest in sicer zelo pogoste (posredovalniški strežniki, čebulasto usmerjanje), pogoste (spremenljivi naslovi, slepo opuščanje, usmerjanje mesh in mrežno usmerjanje) in redko uporabljene (generiranje umetnega prometa). Na podlagi analize vseh šestih mehanizmov smo lahko ugotovili, da se ne razlikujejo le po načinu delovanja, ampak tudi po vrstah anonimnosti, ki jih podpirajo. Posredovalniški strežniki največkrat zagotavljajo anonimnost pošiljatelja, čebulasto usmerjanje anonimnost pošiljatelja, prejemnika in povezave. Za spremenljive naslove je značilna anonimnost pošiljatelja, za slepo opuščanje anonimnost pošiljatelja in prejemnika ter za usmerjanje mesh in mrežno usmerjanje anonimnost pošiljatelja, prejemnika in povezave. Generator umetnega prometa pa zagotavlja anonimnost povezave.

V nadaljevanju smo si pogledali, kako je z anonimnostjo po modelu ISO/OSI. Pogledali smo si delovanje vsake plasti posebej, njihove glavne naloge in na podlagi tega smo ugotavljali, ali res zagotavlja katero izmed vrst anonimnosti, če sploh katero. Ugotovili smo, da ima anonimnost največjo vlogo na tretji, to je omrežni, plasti.

Nato smo natančneje analizirali protokol SOCKS. To pomeni, da smo si ogledali njegovo delovanje, kje se ga uporablja, kakšne prednosti in slabosti

ima, nato pa smo natančno opisali obe verziji. Pri tem smo se osredotočili na izgled paketov in samega pošiljanja paketov. Posebej nas je zanimalo rokovanje pri obeh verzijah. Ugotovili smo, da so rokovanja zelo različna, a vendar povezana. Namreč, če pogledamo rokovanje protokola SOCKS5 za promet TCP in hkrati še rokovanje protokola SOCKS4 opazimo, da ima SOCKS5 v bistvu rokovanje protokola SOCKS4 z dodatnimi pozdravnimi paketi in paketi za avtentikacijo. Pri analizi smo ugotovili, da posredovalniški strežnik SOCKS5 zagotavlja anonimnost pošiljatelja, v primeru veriženja večih posredovalniških strežnikov SOCKS5 pa zagotavlja še anonimnost prejemnika in povezave.

Pri praktičnem delu diplomske naloge smo se osredotočili na postavljanje posredovalniškega strežnika SOCKS5, vzpostavljanje tunela SSH in kriptiranje prometa skozi tunel. Videli smo, da je način res preprost in ne zahteva veliko znanja. Razmislili smo tudi o nadgradnjah. Izvedeni način bi lahko uporabili skupaj s spremenljivimi naslovi, da bi dosegli anonimnost pošiljatelja. Lahko bi tudi spremenili protokol SOCKS5, da bi, poleg zagotavljanja anonimnosti pošiljatelja, imel vgrajeno tudi kriptiranje prometa.

Skozi diplomsko nalogo se je izkazalo, da je anonimnost in varnost na internetu še vedno velik problem. Uporabniki interneta se ne zavedajo vseh možnih nevarnosti in v trenutku so lahko žrtev spletnih kriminalcev. S preprostim načinom, kot je prikazan, je možno zagotoviti dodaten nivo varnosti. Tak način uporablja precej uporabnikov interneta in vsi so z njim zadovoljni. S tem zmanjšamo stopnjo nevarnosti oziroma tveganja in tako lahko skoraj brezskrbno deskamo po spletu. Kljub temu pa se določenih napadov s tem ne moremo ubraniti.

Literatura

[1] G. Ferro (2008) "Fast Introduction to SOCKS Proxy", dostopno na: http://etherealmind.com/fast-introduction-to-socks-proxy/.

[2] C. Hoffman (2012) "VPN vs. SSH Tunnel: Which Is More Secure?", dostopno na: www.howtogeek.com/118145/vpn-vs.-ssh-tunnel-which-is-more-secure/.

[3] A. Jones (2004) "Anonymous Communication on the Internet", dostopno na: www10.csse.rose-hulman.edu/Papers/Jones.pdf.

[4] N. Krawetz, "Introduction to Network Security", Charles River Media, 2006, str. 111, 149, 199, 255-275, 277.

[5] Y.D. Lee (2012) "SOCKS: A protocol for TCP proxy across firewalls", dostopno na: www.openssh.org/txt/socks4.protocol.

[6] Y.D. Lee (2012) "SOCKS 4A: A Simple Extension to SOCKS 4 Protocol", dostopno na: www.openssh.org/txt/socks4a.protocol.

[7] M. Leech et al (1996) RFC 1928 - "SOCKS Protocol Version 5", dostopno na: www.ietf.org/rfc/rfc1928.txt.

[8] M. Leech (1996) RFC 1929 - "Username/Password Authentication for SOCKS V5", dostopno na: www.ietf.org/html/rfc1929.

[9] A. Pfitzmann, M. Hansen (2010) "A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability,

Unobservability, Pseudonimity and Identity Managment", dostopno na: http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.32.doc.

[10] L. Pimenidis, T. Kölsch, "Transparent Anonymization of IP Based Network Traffic", predstavljeno na: *10th Nordic Workshop on Secure IT-systems*, Tartu, Estonija, 2005.

[11] D. Reed, "Applying the OSI seven-layer model to Information Security", SANS Institute, 2003.

[12] M. G. Reed, P. F. Syverson, D. M. Goldschlag, "Proxies for Anonymous Routing", v zborniku: Computer Security Applications Conference, 1996, str. 95-104.

[13] Wikipedia, Anonymity, posodobljeno: 13.8.2012 10:27, pridobljeno: 13.8.2012 21.55.

[14] Wikipedia, Anonymous web browsing, posodobljeno: 12.8.2012 03:27, pridobljeno: 4.9.2012 22:40.

[15] Wikipedia, Mesh networking, posodobljeno: 25.8.2012 15:37, pridobljeno: 7.9.2012 00:50.

[16] Wikipedia, Network Topology, posodobljeno: 5.9.2012 13:49, pridobljeno: 7.9.2012 00:48.

[17] Wikipedia, Onion routing, posodobljeno: 3.7.2012 20:10, pridobljeno: 3.7.2012 22:35.

[18] Wikipedia, SOCKS, posodobljeno: 9.9.2012 9:05, pridobljeno: 10.9.2012 10:22.

[19] Wikipedia, Tor (anonymity network), posodobljeno: 31.8.2012 9:38, pridobljeno: 7.9.2012 15:35.

[20] B. S. Wilson (2003) "What is SOCKS?", dostopno na: http://infosecwriters.com/texts.php?op=display&id=65.

[21] "IP address, Proxy, Socks4 & Socks5", dostopno na: www.bustathief.com/ip-address-proxy-socks4-socks5/.