Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matjaž Marolt

Protokoli za zagotavljanje zasebnosti z uporabo steganografije

diplomsko delo

univerzitetni študij računalništva in informatike

Mentor: prof. dr. Denis Trček

Ljubljana 2013

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

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: 01933/2013
Datum: 08.08.2013

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

Kandidat: **MATJAŽ MAROLT**

Naslov: **PROTOKOLI ZA ZAGOTAVLJANJE ZASEBNOSTI Z UPORABO STEGANOGRAFIJE**

STEGANOGRAPHY BASED PROTOCOLS FOR PRIVACY PROVISIONING

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Zasnujte in implementirajte rešitve za zagotavljanje zasebnosti v protokolnem skladu TCP/IP in sicer tako, da uporabite steganografske metode. Omenjene rešitve tudi ustrezno evalvirajte s stališča zaščite zasebnosti in pa njihovih performans.

Mentor:

prof. dr. Denis Trček

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matjaž Marolt, z vpisno številko **63080160**, sem avtor diplomskega dela z naslovom:

Protokoli za zagotavljanje zasebnosti z uporabo steganografije

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Denisa Trčka,
- 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 12. avgusta 2013 Podpis avtorja:

Zahvalil bi se vsem, ki so mi kakorkoli pomagali pri izdelavi diplomske naloge, predvsem pa mentorju, kolegom s fakultete, staršem in dekletu.

Kazalo

Seznam uporabljenih kratic in okrajšav

ISO - International Organization for Standardization
OSI - Open Systems Interconnection
TCP - Transmission Control Protocol
IP - Internet Protocol
CWR - Congestion Window Reduced
URG - Urgent
PSH - Push
ACK - Acknowledge
PSH - Push
RST - Reset
SYN - Synchronize
FIN - Finish
IPSec - Internet Protocol Security
NIDS - Network Intrusion Detection System
BASE - Basic Analysis and Security Engine
XOR - Exclusive OR
ASCII - American Standard Code for Information Interchange

Povzetek

Živimo v dobi zelo razširjenih računalniških komunikacij. Z razvojem tehnologije je postalo pošiljanje več tisoč znakov na sekundo zelo enostavno, na žalost pa je enostavno tudi vohunjenje za sporočili, ki so namenjena tretji osebi. Zagotavljanje zasebnosti sporočil je v številnih okoliščinah ključnega pomena. Da to dosežemo, so potrebni premišljeni pristopi, trenutno pa je med njimi najbolj razširjena kriptografija. Kot bomo videli v tem diplomskem delu, lahko v ta namen uspešno uporabimo tudi prikrito pisanje ali steganografijo. V delu bom sprva predstavil splošne značilnosti steganografije, nato pa bom opisal zasnovo in implememtacijo družine protokolov za zagotavljanje zasebnosti v računalniških komunikacijah z uporabo steganografije. Ob tem bom opisal tudi različne steganografske tehnike, ki so jih različni avtorji že razvili in povzel njihove prednosti ter slabosti. Na koncu bom evalviral še svoje protokole, ki sem jih zasnoval in implementiral v tem delu.

Ključne besede: komunikacija, TCP/IP, zasebnost, steganografija

Abstract

We are living in an era of computer communication. The development of technology has rendered sending thousands of characters per second very easy. Unfortunately, spying on the messages intended for a third person has also become easier. Ensuring message privacy is often crucial. To achieve it, however, it is necessary to apply cautious approaches, and cryptography is currently the most widely used. The thesis shows that concealed writing or steganography can also be successfully adopted in this respect. The thesis first outlines the basic features of steganography, followed by a presentation of the design and implementation of a protocol group for ensuring privacy in computer communication through steganography. In addition, the thesis describes various steganography techniques that have already been developed by different authors and points to their advantages and disadvantages. It concludes with an evaluation of the protocols we designed and implemented for the purpose of the thesis.

Key words: communication, TCP/IP, privacy, steganography.

Poglavje 1

Uvod

Ker je naslov dela *Protokoli za zagotavljanje zasebnosti z uporabo steganografije*, bi bilo v ta namen treba najprej definirati, kaj pomeni zasebnost in kaj steganografija.

Zasebnost pomeni, da lahko poslano sporočilo prebere le točno določen naslovnik. Težava pri računalniških komunikacijah je slednja. Pošiljatelj pošlje sporočilo v informacijski kanal. Izraz informacijski kanal sem uporabil zato, ker ne želim govoriti o podrobnostih medija, po katerem se sporočilo prenese, saj gre tu za skupek naprav, kot so: operacijski sistem, mrežna kartica, kabel, brezžični oddajnik in sprejemnik, usmerjevalnik... Pomemben vidik pri informacijskem kanalu računalniških komunikacij je ta, da zagotavlja prenos podatkov med dvema točkama in da lahko v večini primerov podatke, ki potujejo po kanalu, vidi tudi oseba, ki ni naslovnik sporočila. Tej osebi ponavadi pravimo napadalec. Delo napadalca si lahko predstavljamo kot na primer poštnega uslužbenca, ki prebira vsa pisma iz glavnega poštnega nabiralnika, čeprav so le-ta naslovljena na tretje osebe. Prav tako je možno, da napadalec ponaredi svoj naslov in se tako lažno izdaja za naslovnika sporočila. V tem delu se s podrobnostmi o dostopu napadalca do vsebine sporočila v informacijskem kanalu ne bom ukvarjal, pomembno je le, da vemo, da je to možno in da moramo zasebnost zagotoviti na drugačen način.

Najbolj razširjen način za zagotavljanje zasebnosti v današnjem času je kriptografija. Pri kriptografiji gre za postopek, kjer sporočilo pred pošiljanjem spremenimo do neprepoznavnosti oz. ga zakriptiramo. Tako v primeru, če napadalec prebere sporočilo v informacijskem kanalu, zasebnost še vedno ohranimo, saj napadalec spremenjenega (kriptiranega) sporočila ne bo znal razvozlati. Moramo pa seveda zagotoviti, da bo pravi naslovnik z razliko od napadalca kriptirano sporočilo lahko dobil nazaj v berljivo obliko oz. ga dekriptiral. To dosežemo s pomočjo deljenih skrivnosti med pošiljateljem in naslovnikom. O podrobnostih kriptografije prav tako v tem delu ne bom pisal. Pomembno je le, da poznamo osnovni koncept kriptografije. Kriptiranje in dekriptiranje sporočil je računsko precej zahtevno ter temelji na zapletenih matematičnih algoritmih.

Zasebnost pa lahko zagotovimo tudi z uporabo prikritega pisanja oz. steganografije. Medtem, ko kriptografija skrije pomen sporočila, steganografija njegov obstoj. Namesto da napadalec v kanalu prebere neprepoznavno sporočilo, le-ta v primeru steganografije sporočila sploh ne opazi. Vendar pa obstoja sporočila ne moremo kar tako prikriti. Uporabiti moramo namreč 2 sporočili. Prvo sporočilo je daljše in nezasebno, drugo sporočilo pa krajše in zasebno. Dele drugega, zasebnega sporočila s pomočjo različnih tehnik pazljivo skrijemo v prvo sporočilo. Napadalec bo zmotno mislil, da po kanalu prenašamo zgolj prvo sporočilo, pravi naslovnik pa bo znal iz tega sporočila izluščiti tudi skrito zasebno sporočilo. Pri tem moramo biti zelo pazljivi, da vidno ne pokvarimo prvega sporočila, saj mora le-to napadalcu izgledati kot pristno in nespremenjeno. Steganografija je za razliko od kriptografije ponavadi računsko nezahtevna, saj gre tu v mnogih primerih za zelo preproste operacije, kot je na primer sprememba podatkov na določenih mestih.

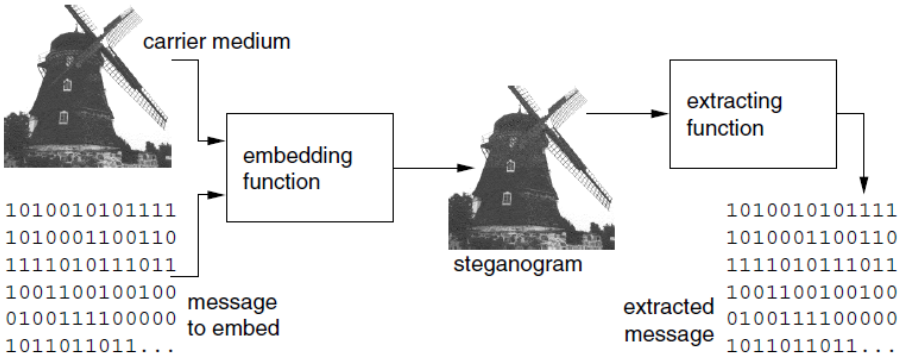
V tem delu sem zasnoval in implementiral družino protokolov za pošiljanje zasebnih sporočil med računalniki s kombinacijo različnih steganografskih tehnik.

Poglavje 2

Steganografija

Beseda steganografija izhaja iz grščine in pomeni prikrito pisanje, razvila pa se je že mnogo pred računalniškimi komunikacijami. Grki so takrat prenašali zaupna sporočila tako, da so sužnjem na obrito glavo vtetovirali sporočilo. Ko so sužnju zrasli lasje, so ga poslali do naslovnika, ki mu je ponovno obril glavo in prebral sporočilo. Več kot 1000 let pozneje se je steganografija začela uporabljati tudi v računalništvu s sicer povsem drugačnimi tehnikami, toda osnovni princip ostaja: zakritje obstoja sporočila.

Zasebno sporočilo je v računalništvu sicer podobno tistemu iz zgodovine. Gre namreč za zaporedje znakov. Vendar se tu za zakritje sporočila poslužujemo tehnik, ki se precej razlikujejo od tetoviranja in striženja las. Vse izmed njih delujejo na načine, s katerimi zasebno sporočilo skrijemo v daljše, nezasebno sporočilo. Tehnike se razlikujejo tako v tipu nezasebnega sporočila, kot tudi samih podrobnostih o tem, kako skriti sporočilo znotraj drugega. Najenostavnejša je tehnika zamenjave določenih znakov nezasebnega sporočila z znaki iz zasebnega sporočila, vendar s tem tvegamo vidno spremembo v nezasebnem sporočilu, kar lahko takoj vzbudi sum pri napadalcu. Zelo pomembna je torej tudi izbira tipa nezasebnega sporočila. Če na primer skrivamo tekst znotraj teksta tako, da zamenjamo vsak deseti znak nezasebnega sporočila z znakom zasebnega sporočila, bo napadalec hitro ugotovil, kateri znaki so bili zamenjani in tako tudi razvozlal zasebno



Slika 2.1: Skrivanje zasebnih podatkov v sliko

sporočilo. Zaradi tega je najbolj priljubljen tip podatkov za steganografijo multimedija, znotraj katere lahko skrijemo poljuben tip podatkov, kar prikazuje Slika 2.1. Multimedija namreč vsebuje veliko redundantnih podatkov, ki jih lahko spremenimo, napadalec pa sprememb ne bo opazil. Po drugi strani pa tip zasebnih podatkov niti ni bistven, saj na koncu le-te vedno zapišemo kot približno naključno zaporedje bitov. Pomembno je le, da naslovnik ve, za kakšen tip podatkov gre, saj jih le tako lahko združi v smiseln pomen.

Za dodatno varnost lahko steganografijo tudi kombiniramo s kriptografijo. Zasebno sporočilo tako najprej zakriptiramo, nato pa še skrijemo znotraj drugega sporočila. Možna je tudi uporaba deljene skrivnosti med pošiljateljem in naslovnikom neposredno za steganografijo in brez kriptografije. Tako lahko na primer v deljeno skrivnost vključimo kar samo tehniko skrivanja podatkov. Poleg tega lahko podatke skrijemo samo na peščico izmed vseh razpoložljivih mest za skrivanje in seznam mest, kjer so skriti podatki, vključimo v deljeno skrivnost in podobno. Uporaba deljene skrivnosti nam zagotavlja zasebnost tudi v primeru, da napadalec ve za uporabo steganografije in obstoj sporočila znotraj drugega, ob tem pa vendarle obidemo računsko zahtevnost kriptografije.

2.1 Kvaliteta steganografskih tehnik

Ker je znotraj steganografije razvitih mnogo tehnik, je dobro preveriti, katere so lastnosti za njihovo ocenjevanje uspešnosti. Avtorji iz literature navajajo različno število parametrov uspešnosti, v večini pa omenjajo 4 glavne parametre. To so:

- kapaciteta;
- varnost;
- robustnost;
- računska nezahtevnost.

2.1.1 Kapaciteta

Izraz **kapaciteta** se sliši malce nenavadno, saj ne vemo, na kaj točno se navezuje. Izraz sem uporabil, ker ga uporabljalo tudi drugi avtorji v literaturi. Gre za kapaciteto nosilca podatkov, preko katerega lahko prenesemo zasebno sporočilo. Če bi skrivali podatke v sliko, bi torej govorili o številu bajtov, ki jih lahko v sliki zamenjamo z zasebnimi podatki. To bi bila v tem primeru naša kapaciteta nosilca podatkov. Želimo si imeti čim večjo kapaciteto, saj lahko tako znotraj fiksne dolžine nezasebnih podatkov skrijemo čimveč zasebnih podatkov ali pa fiksno dolžino zasebnih podatkov lahko skrijemo med čim manjšo dolžino nezasebnih podatkov. Nezasebne podatke namreč naslovnik tako ali tako ponavadi zavrže, saj so le-ti namenjeni zgolj za maskiranje zasebnih podatkov. Večja količina nezasebnih podatkov zato predstavlja zgolj nepotreben strošek in časovno zakasnitev.

2.1.2 Varnost

Z vsakim pošiljanjem zasebnega sporočila, skritega z uporabo steganografije, obstaja tveganje, da bo napadalec odkril steganografsko tehniko in s tem tudi

zasebno sporočilo. Pri kriptografiji se navadno zanašamo na to, da napadalec zaradi zapletenosti kriptografskih algoritmov ne bo imel dovolj računske moči, da bi lahko prišel do prave vsebine sporočila v realnem času. Pri steganografiji je pristop povsem drugačen. Tu se moramo zanašati zgolj na to, da napadalec ne bo vedel za obstoj skritega, zasebnega sporočila. V primeru, da napadalec odkrije kakršnekoli nepravilnosti v nezasebnem sporočilu in tako posumi na uporabo steganografije, je zasebno sporočilo bistveno bolj podvrženo nevarnosti odkritja. Varne tehnike steganografije so zato tiste, ki čim manj spreminjajo izgled nezasebnega sporočila in tako veliko težje vzbudijo sum pri napadalcu.

2.1.3 Robustnost

Tretji izmed glavnih parametrov za ocenjevanje uspešnosti steganografske tehnike je robustnost. Raven robustnosti nam pove, kako dobro se zasebno sporočilo ohranja v nenavadnih okoliščinah. Denimo, da napadalec zasebnega sporočila ne zmore razvozlati, obenem pa se zaveda, da skrito zasebno sporočilo znotraj nezasebnega obstaja. Napadalec lahko še vedno spremeni nekaj podatkov v nezasebnem sporočilu in kljub temu ohrani njegov zunanji izgled. Vendar pa s tem dejanjem napadalec zelo verjetno pokvari vsaj nekaj podatkov v zasebnem sporočilu, pošiljatelj in naslovnik pa se tega ne bosta zavedala, zato bo naslovnik prejel spremenjeno in najverjetneje neuporabno zasebno sporočilo, tega pa se ne bo niti zavedal. Visoka robustnost v tem primeru pomeni, da je steganografska tehnika takšna, da je zelo težko spremeniti ali uničiti samo zasebno sporočilo, nezasebnega pa ob tem ohraniti. Napadalcu še vedno ostane možnost spremembe obeh sporočil hkrati, vendar bo v tem primeru tudi naslovnik zaznal, da je s sporočilom nekaj narobe. Visoka robustnost je zato vedno zaželjena.

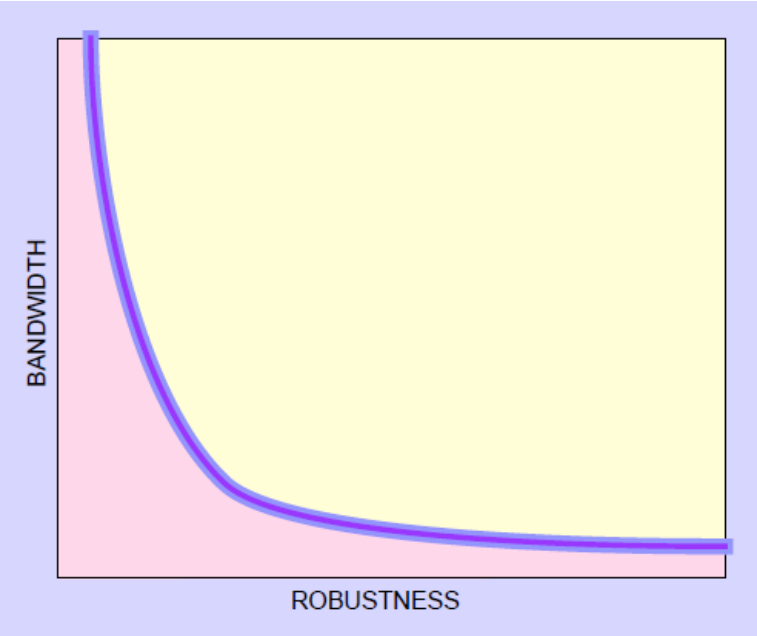
2.1.4 Računska nezahtevnost

Podobno kot pri kriptografiji obstajajo tudi pri steganografiji bolj in manj zapletene tehnike. Nasploh velja, da je izvajanje steganografije na računalniku računsko zelo nezahtevno. V veliki večini se izvaja zgolj zamenjava podatkov na točno določenih mestih. Vendar pa so tudi nekatere steganografske tehnike precej bolj zapletene. Ta zapletenost ponavadi prinese boljše rezultate v smislu kapacitete, varnosti in robustnosti, vendar pa zahteva tudi veliko več računanja in s tem računalniških resursov. Računski zapletenosti se želimo izogniti, saj le-ta ponavadi prav tako predstavlja nepotreben strošek in morebitne dodatne zakasnitve. Še posebej je računska nezahtevnost pomembna v senzorskih omrežjih, saj je tam večina računalnikov napajana preko baterije, zapleteno računanje pa porabi veliko energije in s tem močno skrajša dobo delovanja računalnika oz. senzorja.

2.2 Preference steganografskih tehnik

Pred izbiro steganografske tehnike se moramo odločiti, kaj od nje sploh pričakujemo oz. kakšne so naše zahteve. Parametri, s katerimi ocenimo uspešnost tehnik, so sicer znani, vendar se le-ti na žalost med seboj izključujejo. Tako ima na primer tehnika, ki dosega visoko kapaciteto kot slabost nizko varnost. Če torej skrijemo veliko količino zasebnih podatkov med nezasebne, s tem tudi bistveno povečamo tveganje za razkritje le-teh, saj bodo nezasebni v veliki meri spremenjeni, kar bo pri napadalcu lažje vzbudilo sum.

Delo W. Bendra [3] opisuje podobno zvezo med kapaciteto in robustnostjo, kar prikazuje tudi Slika 2.2. Če želimo doseči visoko robustnost, moramo v večini primerov zasebne podatke opremiti z dodatnimi reduntantnimi podatki, kot je na primer kontrolna vsota. Ob zelo visoki robustnosti je lahko dodanih redundantnih podatkov nekajkrat toliko, kot je velikost izvornega zasebnega sporočila. Dodani redundantni podatki morajo prav tako ostati zasebni, zato moramo za ustrezen faktor zmanjšati kapaciteto za prenos iz-



Slika 2.2: Odvisnost med kapaciteto in robustnostjo. Vir: [3]

vornega zasebnega sporočila.

Zelo napredne tehnike lahko dajejo tudi dober rezultat na vseh treh področjih, vendar so redke in ponavadi precej zapletene. Kot sem že omenil, pa imajo zapletene tehnike bistveno večjo računsko zahtevnost, kar jih lahko v danih okoliščinah naredi neuporabne. Zato je bistveno, da se odločimo, katerim parametrom za ocenjevanje uspešnosti želimo dati prednost in na podlagi tega izberemo ustrezne steganografske tehnike.

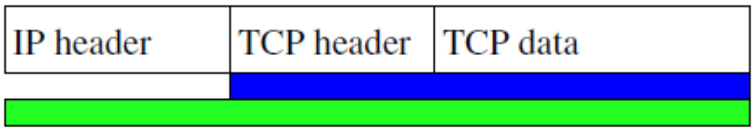
Prav zaradi tega sem se tudi odločil za zasnovo družine protokolov namesto enega samega protokola. Tako lahko namreč glede na okoliščine in osebne preference uporabimo protokol, ki nam v danih okoliščinah najbolj ustreza.

2.3 Steganografija v računalniških komunikacijah

Kot smo že omenili je multimedija idealen tip podatkov za steganografijo zaradi velikega števila redundanc. Zagotovo je to tudi najbolj raziskana veja steganografije. V literaturi najdemo veliko tehnik za skrivanje podatkov tako v slike kot tudi v zvočne in video posnetke.

V tem diplomskem delu pa sem uporabil drugačno vrsto steganografije. Ta za nezasebne podatke ne uporablja multimedije, pač pa kontrolne podatke, ki služijo računalniški komunikaciji. Pri računalniških komunikacijah se moramo držati vseh potrebnih protokolov, nekateri med njimi pa so precej zapleteni. Ob tem si pomagamo s pomočjo omrežnih modelov, kot sta ISO/OSI in TCP/IP. Slednji ima ponavadi definiranih 5 plasti, vsaka plast pa zagotavlja komunikaciji svojo funkcijo. Večina plasti med sporočilo vstavi tudi svoje kontrolne podatke, ki omogočajo plasti delovanje. V tem delu se bom omejil na omrežno in transportno plast, predvsem na protokola TCP in IP, pri čemer TCP deluje na transportni plasti, IP pa na omrežni.

Ob pošiljanju sporočila pošiljateljev računalnik sporočilo razbije na več krajših sporočil, tako imenovanih paketov. Vsak paket nato opremi še s kontrolnimi podatki posamezne plasti oz. protokola na posamezni plasti. Te kontrolne podatke imenujemo glave. Razbitje sporočila na več paketov seveda ni obvezno, v kolikor je sporočilo dovolj kratko za pošiljanje v enem samem paketu. Slika 2.3 prikazuje paket opremljen s kontrolnimi podatki protokolov TCP in IP. V prvem polju so kontrolni podatki protokola IP, sledijo kontrolni podatki protokola TCP, na koncu pa so še dejanski podatki, katere želimo poslati v paketu. Če zgradbo paketa primerjamo s pošiljko klasične pošte, sta glavi v paketu podobni podatkom na kuverti, podatki iz zadnjega polja pa podatkom iz pisma v notranjosti kuverte. V primerjavi s klasično pošto pa je kontrolnih podatkov v računalniških komunikacijah vendarle bistveno več.



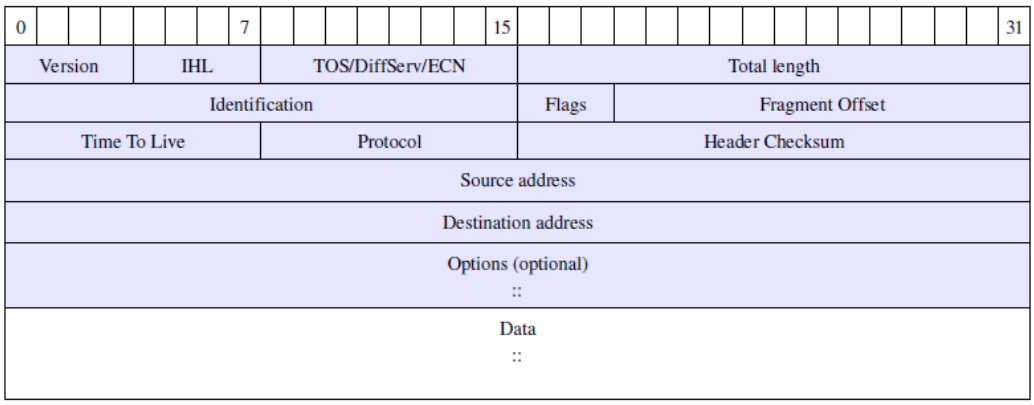
Slika 2.3: Paket opremljen z glavama protokolov TCP in IP

2.3.1 Glava IP

Protokol IP v osnovi skrbi za usmerjanje paketov. Obstajata 2 razširjeni verziji tega protokola, in sicer verzija 4 ter verzija 6. Sama logika protokolov je podobna, vendar verzija 6 uporablja daljše naslove. V diplomskem delu sem raziskal predvsem verzijo 4 (IPv4), o kateri bo v nadaljevanju govorilo to delo.

Slika 2.4 prikazuje glavo protokola IPv4. Standardno je glava dolga 20 bajtov in vsebuje kar precej različnih kontrolnih podatkov:

- **Verzija (Version)** je podatek o verziji protokola IP, v našem primeru je nastavljen na vrednost 4.
- **Dolžina glave IP (IHL)** je podatek o dolžini glave protokola IP, saj je glava lahko različo dolga.
- **Tip storitve (TOS)** je polje, ki je v večini neuporabljeno, včasih pa se uporablja za izražanje prioritete paketa.
- **Skupna dolžina (Total length)** je podatek o skupni dolžini paketa;
- **Identifikacija (Identification)** vsebuje identifikacijsko številko paketa, ki se uporablja v primeru fragmentacije. Pakete lahko dodatno razbijemo na fragmente in pošljemo preko kanala vsak fragment posebej. S pomočjo tega polja naslovnik ve, kateremu paketu pripada prejeti fragment, tako da lahko fragmente združi nazaj v celoten paket. Vsi fragmenti iz istega paketa imajo torej enako identifikacijsko številko.



Slika 2.4: Glava IPv4

- **Zastavice (Flags)** sestavljajo trije biti. Prvi bit je rezerviran in mora biti vedno postavljen na vrednost 0. Drugi bit se imenuje **Ne fragmentiraj (Do not fragment)**. Če je ta bit postavljen na 1, potem se tega paketa ne sme fragmentirati, ampak se ga mora v primeru, da je prevelik in je fragmentacija potrebna, enostavno zavreči. Če je ta bit postavljen na 0, potem je fragmentacija dovoljena. Tretji bit se imenuje **Več fragmentov (More fragments)**. Če je le-ta postavljen na 1 pomeni, da to ni zadnji fragment v paketu in da mora naslovnik počakati še prejem preostalih fragmentov. Bit postavljen na 0 pomeni, da je to zadnji fragment iz paketa in da lahko naslovnik takoj sestavi fragmente v paket.

- **Odmik fragmenta (Fragmet offset)** ponazarja pozicijo prvega bajta trenutnega fragmenta v pripadajočem paketu. Prvi fragment tako vsebuje vrednost 0.

- **Preostali čas (Time to live)** je vrednost, ki se odšteje za 1 vsakič, ko paket preide vmesno vozlišče (usmerjevalnik). Ko vrednost pade na 0, se paket zavrže. Pošiljatelj lahko s to vrednostjo nastavi maksimalno število vmesnih vozlišč, ki jih paket lahko prečka na poti do naslovnika.

- **Protokol (Protocol)** vsebuje podatek o tem, kakšen protokol paket

uporablja na transportni plasti.

- **Kontrolna vsota glave (Header checksum)** je izračunana kot 16-bitni eniški komplement eniškega komplementa vsote vseh 16-bitnih besed glave. S pomočjo te vrednosti lahko ugotovimo, če je med prenosom paketa prišlo do spremembe podatkov v glavi. Ko usmerjevalnik paketu zmanjša vrednost Preostali čas za 1, ob tem tudi ponovno izračuna in popravi kontrolno vsoto.

- **Izvorni naslov (Source address)**.

- **Ponorni naslov (Destination address)**.

Glava IP lahko vsebuje tudi dodatne, opcijske kontrolne vrednosti.

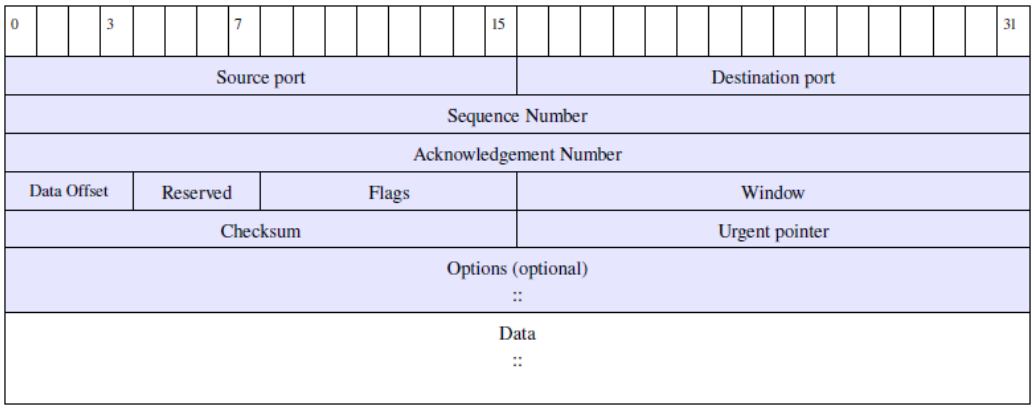
2.3.2 Glava TCP

Protokol TCP v osnovi skrbi za to, da so paketi do naslovnika zares prispeli, in da so prispeli v pravilnem vrstnem redu. Glavo protokola TCP prikazuje Slika 2.5. Obenem pa protokol naslovom doda še vrata. Tako imamo lahko med dvema računalnikoma odprtih več hkratnih navideznih povezav, kjer se preko vsake vsake povezave prenaša svoj tip podatkov. To enostavno dosežemo tako, da pri pošiljanju podatkov uporabljamo različno vrednost vrat. Polja glave TCP so:

- **Izvorna vrata (Source port)**.

- **Ponorna vrata (Destination port)**.

- **Sekvenčna številka (Sequence number)** se uporablja za to, da v toku podatkov ugotovimo, kateri podatki se trenutno pošiljajo. Sekvenčna številka se ob vzpostavitvi povezave določi naključno, nato pa se z vsakim poslanim paketom poveča za število bajtov, vsebovanih v poslanem paketu.



Slika 2.5: Glava TCP

- **Številka potrditve (Acknowledgement number)** nam pove, kateri podatki iz toka podatkov so bili uspešno prejeti.
- **Odmik podatkov (Data offset)** vsebuje dolžino glave TCP v 32-bitnih besedah.
- **Rezervirano (Reserved)** polje je rezervivano za prihodnost in se ga ne uporablja.
- **Zastavice (Flags)** sestavlja 8 bitov. Ti so:
  - **CWR** za nadzorovanje zastojev;
  - **ECE** za nadzorovanje zastojev;
  - **URG** za upoštevanje **Urgentnega kazalca**;
  - **ACK** za potrditev prejetih podatkov;
  - **PSH** za označevanje, da je treba podatke direktno posredovati višji plasti mimo medpomnilnika;
  - **RST** za ponastavljanje komunikacije;
  - **SYN** za inicializacijo komunikacije;
  - **FIN** za ponazoritev, da je pošiljatelj končal s pošiljanjem.

- **Okno (Window)** vsebuje število bajtov, ki jih je računalnik pripravljen prejeti v naslednjem paketu.
- **Kontrolna vsota (Checksum)** celotnega paketa, izračunana pa je z enako metodo kot tista iz glave IP.
- **Urgentni kazalec (Urgent pointer)** vsebuje vrednost kazalca na nujne podatke. Vrednost je relativna glede na začetek podatkov v paketu. Nujni podatki imajo višjo prioriteto pri izvrševanju.

Opcijsko lahko, podobno kot pri protokolu IP, tudi tukaj uporabimo še dodatne kontrolne podatke.

2.3.3 Pregled steganografskih tehnik

Kot vidimo, imamo v glavah različnih plasti modela TCP/IP veliko kontrolnih podatkov. Vsi izmed njih (z izjemo rezerviranih polj) so za zagotavljanje vseh storitev protokola obvezni. Zato si ne moremo privoščiti enostavne zamenjave nekaterih kontrolnih podatkov s podatki iz zasebnega sporočila, ki ga želimo skriti v paket. Toda opazimo tudi, da vsi kontrolni podatki niso potrebni v vseh okoliščinah in prav zato komunikacije po modelu TCP/IP omogočajo tudi številne možnosti za steganografijo znotraj komunikacijskih protokolov samih. V nadaljevanju bom na kratko predstavil nekaj steganografskih tehnik znotraj protokola IP, nato pa še znotraj protokola TCP. Razvite so tudi tehnike za nekatere druge protokole, mnoge od teh tudi na drugih plasteh modela TCP/IP. Kljub temu pa sem se v tem delu omejil na tehnike protokolov TCP in IP, ki so tudi najbolj razvite in ponujajo največ možnosti za steganografijo.

Delo K. Ahsana [2] opisuje uporabo zastavice **Ne fragmentiraj** za skrivanje sporočila. Prepričati se moramo, da do fragmentacije paketa ne bo prišlo. Nato lahko zastavico poljubno spreminjamo za vsak paket posebej in tako skrijemo 1 bit podatkov v vsak paket. Ker zastavica v visokem stanju zgolj prepove fragmentacijo, v paketih, kjer fragmentacija ni potrebna, vrednost zastavice nima vpliva na komunikacijo. Podobno lahko manipuliramo

tudi zastavico **Več fragmentov** in tako skupaj skrijemo 2 bita zasebnih podatkov znotraj paketa. Če želimo skriti več podatkov, pa lahko ob enakih pogojih skrijemo podatke (13 bitov) tudi kar direktno v polje **Odmik fragmenta**, saj tudi to ostane brez pomena, kadar do fragmentacije ne pride.

Podatke lahko skrijemo tudi direktno v polje **Identifikacija** glave IP. Vrednost tega polja namreč ni bistvena za delovanje. Pomembno je le, da imajo vsi fragmenti iz istega paketa v tem polju enako vrednost. Tako lahko v identifikacijsko polje skrijemo 16 bitov podatkov znotraj enega paketa.

Delo S. Zandra [8] opisuje tudi skrivanje podatkov v polje **Preostali čas**. Maksimalna vrednost polja je 255, vendar lahko v skorajda vseh okoliščinah komunikacij paket prispe do naslovnika preko bistveno manjšega števila vozlišč. Privzeto je ta vrednost na večini razširjenih operacijskih sistemov nastavljena na 32–64. V primeru, da sta računalnika, ki kominucirata med seboj soseda in nimata vmesnih usmerjevalnikov, se vrednost polja med prenosom sporočila ne spremeni in lahko vanj direktno skrijemo 8 bitov podatkov, podobno kot pri **identifikacijskem** polju. Tudi v primeru, da zgolj poznamo število vmesnih usmerjevalnikov, lahko skrijemo več bitov podatkov v to polje. A ponavadi ne vemo, koliko usmerjevalnikov bo paket prešel, oz. lahko to le približno ocenimo. Še vedno pa nam tako ostane možnost skrivanja podatkov na primer 1 bit na paket. To dosežemo tako, da za visoko stanje bita postavimo vrednost **Preostali čas** na maksimalno (255), za nizko stanje bita pa na primer privzeto vrednost.

Zasebne podatke lahko skrijemo celo v **Izvorni naslov**. V tem primeru v izvorni naslov namesto svojega naslova vstavimo kar podatke, ki jih želimo skriti. Tako lahko pošljemo 16 bitov podatkov znotraj enega paketa. Možno je celo skrivanje podatkov znotraj **ponornega naslova**, vendar mora v tem primeru naslovnik prestrezati in pregledovati vse pakete na liniji, tudi tiste, ki mu niso namenjeni.

Pristopi, ki sem jih naštel do sedaj, so zelo enostavni. Izvajajo zgolj zamenjavo točno določenih podatkov s podatki iz zasebnega sporočila, kar je tudi najbolj razširjena tehnika steganografije. C. Abad pa je v svojem delu [1]

razložil precej bolj zapleten pristop za skrivanje podatkov. Predlagal je skrivanje podatkov v kontrolno vsoto glave IP, vendar ne direktno, temveč preko precej bolj zapletenih matematičnih operacij. Tehnika sicer posredno skrije 16 bitov podatkov v kontrolno vsoto, vendar pa to samo po sebi ni možno. Algoritem zahteva, da moramo imeti kontrolo nad enim 16-bitnim podatkom drugje v glavi. Avtor predlaga, da uporabimo v glavi dodatno opcijsko polje in tam poljubno nastavimo 16-bitno besedo za potrebe algoritma.

Lahko pa uporabimo tudi pristope, ki skrivajo sporočila povsem iz drugega zornega kota. Primer so pristopi na osnovi časovnosti, kar opisuje na primer delo S. Cabuka [4]. Pristop deluje na podlagi sinhronizacije pošiljatelja in naslovnika. Nato definiramo še časovni interval. Če v tem intervalu pošiljatelj naslovniku pošlje paket, to naslovnik interpretira kot poslano stanje 1. Če v tem intervalu pošiljatelj paketa ne pošlje, pa to pomeni stanje 0. Tako dosežemo hitrost pošiljanja zasebnih podatkov 1 bit na časovni interval. Podobno lahko zasebne podatke izrazimo tudi s pomočjo časovnega zamika med paketi. Bistveni izziv pri časovnih pristopih je zagotovo časovna sinhronizacija pošiljatelja in naslovnika, saj lahko v nasprotnem primeru naslovnik napačno interpretira podatke.

Glava TCP pa podobno kot glava IP ponuja dodatne možnosti za skrivanje podatkov. Podobno kot pri vstavljanju lažnih naslovov v glavo IP, lahko v glavo TCP vstavimo poljubne vrednosti za **izvorna in ponorna vrata**. Skupaj tako lahko zgolj v številke vrat skrijemo do 32 bitov podatkov na paket.

Sekvenčna številka protokola TCP je 32-bitna in se ob vzpostavitvi povezave določi naključno, nato pa ustrezno povečuje glede na to, kateri del toka podatkov se trenutno pošilja. Zato lahko ob vzpostavitvi povezave naključno vrednost **sekvenčne številke** nadomestimo z 32 biti zasebnega sporočila.

Podobno kot kontrolno vsoto in polje **Odmik fragmenta** glave IP, lahko za skrivanje podatkov uporabimo tudi **kontrolno vsoto** glave TCP in pa polje **Urgentni kazalec**. V **kontrolno vsoto** skrijemo podatke povsem na

enak način, kot pri protokolu IP. Vrednost **urgentnega kazalca** se upošteva samo, kadar je zastavica **URG** postavljena na 1, sicer se ignorira. V večini primerov urgentnih podatkov ne potrebujemo, zato lahko v kombinaciji zastavice **URG**, postavljene na 0, v polje **Urgentni kazalec** skrijemo 16 bitov zasebnih podatkov.

Tudi pri protokolu TCP lahko uporabimo tehnike, ki so nekoliko drugačne od klasičnih. Eno izmed njih je v svojem delu [2] opisal K. Ahsan. Protokol TCP namreč zaradi beleženja sekvenčnih številk podatkov omogoča preurejanje paketov v primeru, da le-ti prispejo do naslovnika v napačnem vrstnem redu. To nam da možnost za uporabo steganografije, saj lahko pošiljatelj namerno zamenja vrstni red pošiljanja paketov. Če pošiljatelj zamenja vrstni red pošiljanja v množici n paketov, lahko s tem izrazi $n!$ različnih vrednosti. To pomeni, da lahko v množici n paketov skrijemo $\log_2 n!$ bitov. Ovira pri tem pristopu je ta, da ne sme priti do dodatnih zamenjav v vrstnem redu paketov med prenosom, kar lahko dosežemo z dolgimi premori med pošiljanjem paketov ali pa se prepričamo, da do tega zaradi topologije omrežja ne more priti. Alternativna možnost je tudi uporaba protokola IPSec (na omrežni plasti), ki že sam po sebi zagotavlja prihod paketov preko omrežja v pravilnem vrstnem redu.

V opcijsko polje glav TCP in UDP pa lahko vključimo še mnoge druge kontrolne podatke, ki nam omogočijo še dodatne možnosti za steganografijo. J. Giffin [5] tako predlaga uporabo **časovnih žigov**. **Časovni žig** je 32-bitna vrednost, enota zanjo pa je nanosekunda. Za nemoteno delovanje lahko uporabimo vsaj spodnjih nekaj bitov za skrivanje podatkov. S tem še vedno zagotovimo, da bo vrednost časovnega žiga v vsakem naslednjem paketu večja od tiste v prejšnjem paketu.

Kot vidimo, je raziskanih metod za steganografijo v omrežjih zares veliko, saj sem na kratko opisal le nekaj ključnih znotraj protokolov TCP in IP. S kombinacijo različnih pristopov lahko tako znotraj povsem običajne komunikacije po modelu TCP/IP v vsakem paketu pošljemo vsaj še 100 bitov skritih podatkov. Na žalost pa imajo skorajda vsi pristopi tudi velike pomanjkljivo-

sti. Napadalec, ki se želi dokopati do zasebnih podatkov, skritih v paketih, lahko brez težav nadzoruje tudi uporabo vseh kontrolnih podatkov.

Uporaba opcijskih kontrolnih podatkov je v običajnih komunikacijah zelo nevsakdanja in se ponavadi izvede le za razhroščevanje in podobno. Uporaba tega polja je zato sama po sebi zelo tvegana, saj lahko takoj vzbudi sum pri napadalcu. Uporaba zastavice **Ne fragmentiraj** omogoča kapaciteto skritih podatkov zgolj 1 bit na paket. Sumljiva je tudi uporaba polja **Odmik fragmenta** v primeru, da fragmentacije nimamo oz. uporaba polja **Urgentni kazalec** ob postavitvi zastavice URG na 0. Napadalec lahko v teh primerih tudi ponastavi omenjene vrednosti in s tem ne vpliva na delovanje nezasebne komunikacije.

Identifikacijsko polje je, kot navajajo nekateri avtorji, pri nefragmentiranih paketih privzeto nastavljeno na 0, zato je vsaka druga vrednost sumljiva. V realnosti sicer temu ni vedno tako in nekateri paketi v tem polju vsebujejo naključne, drugi pa celo zaporedne vrednosti. Sumu pri polju **Identifikacija** se lahko izognemo tako, da paket namerno fragmentiramo, vendar lahko napadalec vrednost polja pri vseh fragmentih iz dotičnega paketa zamenja s poljubno vrednostjo in s tem prav tako ne spremeni delovanja nezasebne komunikacije.

Vrednost **Preostali čas**, ki vidno odstopa od znanih privzetih vrednosti, je prav tako alarm za napadalca, predvsem če le-ta ugotovi, da vrednost močno niha znotraj iste povezave.

Če skrijemo podatke v sam naslov IP, to napadalec vidi, kot da z vsakim paketom inicializiramo novo povezavo iz drugega računalnika. Pošiljanje paketov istemu naslovniku iz veliko različnih naslovov je zelo nenavadno, še bolj nenavadno pa je, da se z vsakega naslova pošlje samo 1 paket. Enako velja tudi za vrata protokola TCP.

Časovni pristopi so po drugi strani veliko varnejši, vendar pa tudi nezanesljivi, saj v računalniških omrežjih pogosto prihaja do nepričakovanih zakasnitev.

Kot vidimo, ima večina zgoraj omenjenih pristopov kar veliko vrzeli na

področju varnosti in robustnosti. V večini primerov lahko namreč napadalec brez večjih naporov odkrije lokacije skritih zasebnih podatkov in le-te tudi prebere. Prav tako lahko namenoma spremeni skrite podatke brez vpliva na nezasebno komunikacijo. Najbolj zaskrbljujoče je dejstvo, da je mogoče večino detekcije sumljivih kontrolnih podatkov precej enostavno implementirati. To dejavnost opravljajo tudi nekateri programi za zaščito pred vdori, usmerjevalniki, proxy strežniki in podobno.

Poglavje 3

Zasnova protokolov

V tem delu sem zasnoval družino treh različnih protokolov za zagotavljanje zasebnosti. Poimenoval sem jih kar *Protokol 1*, *Protokol 2* in *Protokol 3*. Vsi protokoli zagotavljajo zasebnost s pomočjo enostavnih steganografskih tehnik, saj je prav nezahtevnost tista vrlina, ki lahko steganografijo postavi pred kriptografijo.

Prvi protokol, označen s številko 1, je najbolj varen in robusten protokol iz družine. Zasnoval sem ga tako, da je pri njem steganografsko aktivnost s strani nadzornika zelo težko zaznati, saj se paketi bistveno ne razlikujejo od klasičnih. Protokol ima kot slabost nizko kapaciteto nosilca zasebnih podatkov.

Drugi protokol sem zasnoval tako, da dosega bistveno višjo kapaciteto nosilca zasebnih podatkov. Protokol še vedno ohranja razmeroma dobro mero varnosti, čeprav je le-ta bistveno nižja, kot pri prvem protokolu, drugi protokol pa pomeni korak nazaj tudi s stališča robustnosti.

Tretji protokol dosega še malenkost višjo kapaciteto nosilca zasebnih podatkov, vendar je pri tem protokolu varnost precej vprašljiva.

Pri zasnovi protokolov sem za vsak protokol sprva izbral želeno mero varnosti, nato pa skušal s steganografskimi tehnikami doseči čim večjo kapaciteto nosilca zasebnih podatkov. Zame je bilo tako ključnega pomena, da sem se znal postaviti na stran napadalca in odkriti morebitne ranljivosti

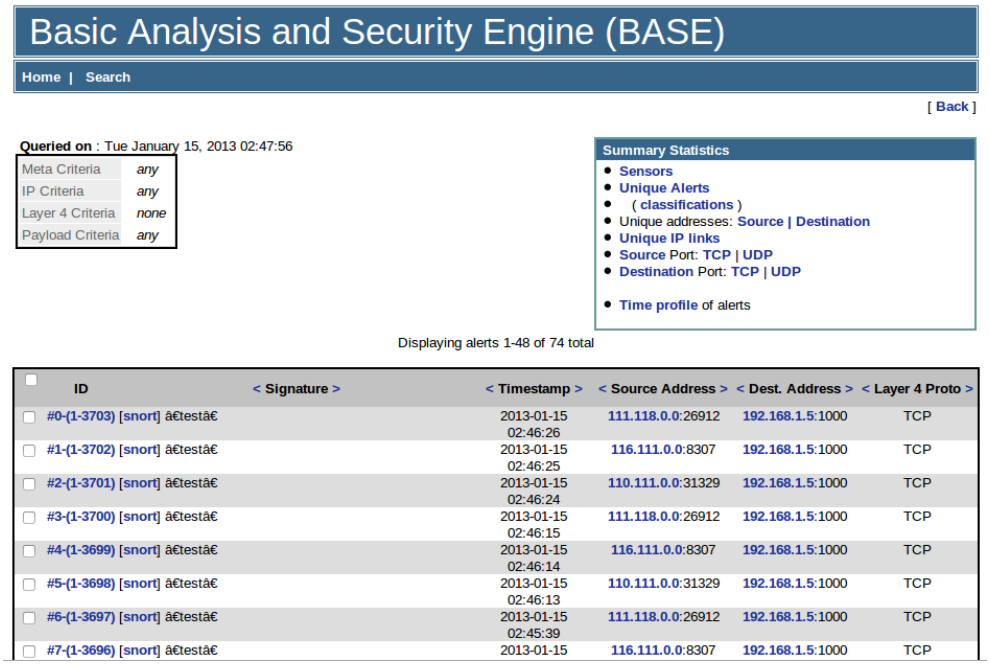
v protokolu. V ta namen sem poizkusil uporabiti tudi programsko orodje *Snort*, ki je namenjeno prav detekciji nepravilnosti in sumljivih podatkov v paketih, ki potujejo po komunikacijskem kanalu.

3.1 Snort

Snort je precej razširjeno programsko orodje, ki služi nadzorovanju omrežnega prometa. Program omogoča več načinov delovanja. Način *Sniffer* tako izpisuje vse pakete, ki se pretakajo po komunikacijskem kanalu, na standardni izhod. Zame je bil predvsem zanimiv način *NIDS (Network Intrusion Detection System)*, saj le-ta v prometu išče zgolj nenavadne, sumljive pakete in uporabnika opozori o ugotovitvah. Steganografija v kontrolnih podatkih paketov je prav takšna dejavnost, ki bi jo način *NIDS* moral odkriti in uporabnika opozoriti o sumljivih paketih. Tako bi lahko z uporabo Snort-a precej objektivno določil stopnjo varnosti svojega protokola.

Snort ima slabost, saj program kot tak ni na voljo z modernim, grafičnim uporabniškim vmesnikom, tako da ga je treba uporabljati prek terminala (konzole) operacijskega sistema, kar zna biti za nadzornika omrežja (ki je hkrati tudi uporabnik programa) precej nepregledno. Vendar tudi za to pomanjkljivost obstaja več rešitev, eno izmed njih pa sem uporabil tudi sam. Poleg Snort-a sem uporabil še grafični vmesnik *BASE (Basic Analysis and Security Engine)*. *BASE* je pravzaprav spletna stran, ki za svoje delovanje potrebuje *Apache* strežnik. Snort nato ugotovitve namesto v terminal izpisuje kar v *MySQL* podatkovno bazo. *BASE* te podatke iz podatkovne baze prebere in jih dinamično predstavi uporabniku na grafično prijazen način.

Slika 3.1 prikazuje izgled spletne strani *BASE*. Na sliki vidimo seznam paketov, ki jih je Snort detektiral kot sporne. Za vsak paket vidimo nekaj osnovnih informacij, kot so opis sporne aktivnosti (v tem primeru test), časovna oznaka, naslovi in vrata pošiljatelja ter naslovnika in podobno. S klikom na paket lahko vidimo več podrobnosti o paketu, prav tako pa nam *BASE* ponuja enostavno sortiranje paketov, filtriranje in podobno.



Slika 3.1: Grafični vmesnik *BASE*

Najprej sem s Snort-om testiral kar varnost tretjega, najmanj varnega protokola. Rezultat me je zelo presenetil. Snort namreč ni zabeležil nobenega spornega paketa kljub temu, da je bila skorajda polovica kontrolnih podatkov zamenjana neposredno z zasebnim sporočilom. Prišel sem do spoznanja, da Snort mogoče vendarle ni optimalno orodje za nadzorovanje komunikacijskega kanala v primeru steganografije. Kljub temu pa Snort omogoča tudi pisanje lastnih pravil za detekcijo spornih paketov, osnovno sintakso pisanja pravil pa opisuje v svojem delu [7] tudi I. Starc.

Poizkusil sem torej še s pisanjem lastnih pravil, a sem kaj kmalu ugotovil, da so pravila bolj namenjena odkrivanju spornih podatkov v vsebini paketov, kot pa kontrolnih podatkov. Ugotovil sem, da nekaterih kontrolnih podatkov s pomočjo pravil ni mogoče preveriti. Kljub temu pa sem želel napisati kar največ možnih pravil, s katerimi bi lahko ocenil varnost svojih protokolov. Koda 3.1 prikazuje eno izmed mojih pravil. Pravilo preverja vse pakete TCP, ki so naslovljeni na naslov 192.168.1.5 (to je naslov naslovnika zasebnega

sporočila) in na vrata 1000, kjer naslovnik prejema zasebna sporočila. Snort nato sproži opozorilo z opisom *Urgent pointer not null with URG flag not set*, če je zastavica URG postavljena na 0 in je vrednost polja Urgentni kazalec različna od 0. S pisanjem lastnih pravil sem imel veliko težav, saj pravila niso dajala rezultatov, ki bi jih po dokumentaciji morala.

```
alert tcp any any -> 192.168.1.5 1000 (flags:!U;urg:!0;\
msg:"Urgent pointer not null with URG flag not set";)
```

Koda 3.1: Lastnoročno dodano pravilo programa Snort

Z lastnimi pravili programa Snort lahko naenkrat preverimo zgolj en sam paket, kar pomeni, da lahko s tem detektiramo le tiste najbolj ranljive pakete. Bolj varne steganografske tehnike pa seveda znajo takšno pregledovanje prelisičiti in je za njihovo detekcijo potreben pregled več paketov. Metode iskanja sumljivih paketov, ki temeljijo na pregledovanju enega samega paketa naenkrat, bom poimenoval *statične*. S pojmom *dinamične* metode pa bom poimenoval tiste, ki delujejo na osnovi shranjevanja paketov v pomnilnik in pregledovanja več paketov naenkrat. Kot smo videli v povzetku steganografskih tehnik, so nekatere izmed njih take, da je posamezen paket videti povsem normalen in brez nepravilnosti, ko pa ga postavimo v okolje nekaj sosednjih paketov, lahko opazimo, da gre pravzaprav za steganografsko aktivnost. Seveda pa je takšno aktivnost bistveno težje odkriti in za to potrebujemo *dinamične* metode iskanja sumljivih paketov, ki so bistveno bolj zapletene in tudi računsko bolj zahtevne.

3.2 Program za detekcijo steganografije v paketih

Zaradi potrebe po zanesljivih statičnih in dinamičnih metodah za detekcijo steganografske aktivnosti sem se odločil, da bom kar sam napisal program, v katerem bom implementiral obe vrsti metod. Program sem napisal v programskem jeziku C.

Najprej sem se moral odločiti, katere statične metode za detekcijo steganografije bom implementiral. Znotraj protokola IP sem implementiral le eno samo preverjanje. Program preveri, ali je ob postavitvi zastavice Ne fragmentiraj v visoko stanje tudi polje Odmik fragmenta postavljeno na 0. V nasprotnem primeru gre lahko v tem polju za steganografijo, na kar nas program opozori.

Znotraj protokola TCP sem implementiral še tri druge statične metode. Ena izmed njih preveri vrednost polja Rezervirano, ki mora imeti v normalnih okoliščinah vrednost 0, sicer je velika verjetnost, da so v njem skriti podatki, na kar program tudi opozori. Druga metoda preverja vrednost polja Številka potrditve ob zastavici ACK v nizkem stanju. Številka potrditve ima v normalnih okoliščinah v takšnem primeru vrednost 0. Podobno program preveri tudi vrednost polja Urgentni kazalec ob zastavici URG v nizkem stanju, kar prikazuje Koda 3.2.

```
if((recv_pkt.tcp.urg == 0) &&
(recv_pkt.tcp.urg_ptr != 0))
{
printf("Alert: Urgent pointer not null with URG flag
not set\n");
fprintf(output, "Alert: Urgent pointer not null with
URG flag not set\n";
}
```

Koda 3.2: Koda, ki opozarja na neničelno vrednost polja Urgentni kazalec, ko je zastavica URG postavljena na 0

Kot vidimo so statične metode tudi zelo enostavne za implementacijo, česar pa za dinamične metode ne moremo trditi. Če želimo kakorkoli statistično zaznati odstopanje množice steganografskih paketov od navadnih, moramo v ta namen v pomnilnik shranjevati veliko število paketov, kar je računsko precej zahtevno in energijsko potratno, nenazadnje pa tudi implementacija ni ravno enostavna, zato se za prave statistične metode znotraj

dinamičnih metod nisem odločil. Implementiral pa sem 3 nekoliko bolj enostavne metode. Prva metoda odkriva raznolikost izvornih naslovov v zadnjih nekaj paketih. Če je ta raznolikost previsoka (na primer dobimo 10 zaporednih paketov iz 10 različnih naslovov), program izpiše opozorilo, saj v tem primeru lahko sumimo na skrivanje zasebnih podatkov znotraj izvornega naslova. Podobno program preverja tudi raznolikost izvornih vrat, ki v normalnih okoliščinah prav tako ne bi smela biti prevelika, preverjanje pa prikazuje Koda 3.3. Tretja dinamična metoda preverja vrednost identifikacijske številke. Če se v nekaj zaporednih nefragmentiranih paketih ista vrednost ponovi lahko sumimo, da vrednosti niso naključne, kot bi morale biti v normalnih paketih.

```
uq_count = 0;
for(j=0 ; j<buf_len ; j++){
        match = 0;
        for(k=0 ; k<uq_count ; k++){
                if(packets[j].tcp.source == uniques[k]){
                        match = 1;
                }
        }
        if(match == 0){
                uniques[uq_count]=packets[j].tcp.source;
                uq_count += 1;
        }
}
if(uq_count >= buf_len-1){
        printf("Alert: %d different source ports
detected in last %d packets\n", uq_count, buf_len);
        fprintf(output,"Alert: %d different source ports
detected in last %d packets\n", uq_count, buf_len);
}
```

Koda 3.3: Koda, ki opozarja na preveliko raznolikost izvornih naslovov v

paketih

Metode, ki sem jih implementiral, še zdaleč niso vse možne metode, s katerimi je mogoče iskati steganografske aktivnosti in tudi niso glavni namen tega dela. Zato sem implementiral le nekaj metod, ki so mi pomagale na varnost mojih protokolov pogledati še iz zornega kota napadalca.

3.3 Protokol 1

Prvi protokol za zagotavljanje zasebnosti sem zasnoval tako, da ga lahko vsaj delno implementiramo tudi v drugačnem implementacijskem okolju. Tako na primer lahko skrijemo nekaj podatkov v paket tudi pri komunikaciji, ki ne poteka po modelu TCP/IP. Uporabil sem nekoliko drugačne steganografske tehnike od najbolj znanih. Razlog je bil predvsem pomanjkanje robustnosti, pa tudi varnosti teh tehnik, kar sem opisal že v prejšnjem poglavju. Protokol je tudi najbolj varen iz družine, zato je zasnovan tako, da njegove dejavnosti ni mogoče (oz. jo je težko) zaznati niti z dinamičnimi metodami za detekcijo steganografije.

3.3.1 Sekvenčna številka

Edina tehnika od omenjenih s strani drugih avtorjev, ki se mi je zdela sprejemljiva za uporabo pri prvem protokolu s stališča varnosti in robustnosti, je skrivanje podatkov v sekvenčno številko ob inicializaciji povezave. Ta se praviloma določi naključno, zato jo lahko nadomestimo z zasebnimi podatki dolžine 32 bitov in s tem ne vzbudimo nikakršnih sumov pri napadalcu. Paziti moramo le, da so zasebni podatki čimbolj naključni, saj lahko napadalec v nasprotnem primeru hitro opazi nepravilnosti, čeprav bi lahko kaj takšnega opazil tudi pri običajnih paketih. Napadalec bi lahko vrednost tudi zamenjal s poljubno naključno vrednostjo, vendar za to nima razloga, če nima sumov o steganografiji. Protokol TCP je zasnovan tako, da mora biti sekvenčna številka ob inicializaciji čim bolj naključna. Dobra naključnost je računsko

precej zahtevna, zato napadalec najverjetneje ne bo brez razloga menjal sekvenčne številke z novo, naključno določeno. Odločil sem se, da to tehniko uporabim v svojem protokolu in tako pridobim 32 bitov zasebnih podatkov v prvem paketu. Tehnika je dobrodošla predvsem pri kratkih zasebnih sporočilih, saj pri velikem številu poslanih paketov po isti povezavi dodana vrednost te metode izgubi svoj pomen.

3.3.2 Izvorna vrata

Ob inicializaciji povezave lahko na podoben način za skrivanje podatkov uporabimo tudi izvorna vrata. Spreminjane izvornih vrat v vsakem paketu posebej bi bilo nenavadno, vendar lahko ob inicializaciji to sicer precej poljubno vrednost zamenjamo s 16 biti zasebnih podatkov. Napadalec v tem primeru nima pravega razloga za sum, prav tako pa ne more številke vrat spremeniti brez vpliva na nezasebno komunikacijo.

3.3.3 Okno

V oči mi je padlo tudi polje Okno glave TCP. V to polje vsak računalnik vstavi število bajtov, ki jih je pripravljen prejeti v naslednjem paketu. V mojem primeru imamo sicer enosmerno komunikacijo, vendar pa je večina komunikacije TCP dvosmerna. Tako pošiljatelj ob pošiljanju paketa naslovniku v ta paket vključi tudi informacijo o tem, koliko podatkov je pripravljen prejeti v naslednjem paketu, ko bo smer komunikacije obrnjena. V primeru enosmerne komunikacije pa je to polje odveč, zato lahko vanj skrijemo 16 bitov zasebnih podatkov. Vrednost polja se lahko spreminja od paketa do paketa tudi pri povsem običajni komunikaciji, zato napadalec težje posumi na steganografijo. Kljub temu pa opazne razlike ostajajo, saj se v vsakdanjih komunikacijah ta vrednost ponavadi ne spremeni v vsakem naslednjem paketu in kar je še bolj pomembno, spremembe niso tako drastične in ne zavzemajo celotnega območja vrednosti, ki ga lahko zavzame 16 bitno število. Zato sem se odločil, da bom znotraj protokola uporabil le spodnjih 8 bitov

za skrivanje podatkov, zgornjih 8 pa postavil fiksno na vrednost, ki je dokaj pogosta v računalniških komunikacijah. Tako bodo vrednosti tega polja nihale nekje okrog zelo pogoste vrednosti, kar je dosti manj vpadljivo od nihanja po celotni zalogi vrednosti. Napadalec si prav tako ne sme privoščiti spremembe vrednosti, saj bi v tem primeru prav tako vplival tudi na nezasebno komunikacijo. Ob obrnjeni komunikaciji bi namreč nov pošiljatelj novemu naslovniku lahko poslal preveč ali premalo podatkov. Pošiljanje premajhnega števila podatkov je zelo neoptimalno, pošiljanje preveč podatkov pa pomeni, da naslovnik teh podatkov zaradi preobremenjenosti ne bi mogel sprejeti in obdelati.

3.3.4 Skupna dolžina

V tem protokolu sem uporabil še četrto tehniko, ki pa z razliko od prejšnjih treh deluje na protokolu IP in je zelo splošna. Protokol IP nam omogoča pošiljanje paketov poljubne dolžine, kar se navezuje tudi na polje Okno protokola TCP v prejšnjem odstavku. Tudi v vsakdanji komunikaciji je dolžina poslanih paketov precej nekonstantna. To pomeni, da lahko skrijemo podatke tudi v polje Skupna dolžina glave IP in posledično spreminjamo dolžino paketa. Vendar pa je tudi tu problem varnosti podoben kot pri polju Okno. Pogosta nihanja so nevsakdanja, še bolj nevsakdanje pa je nihanje vrednosti po celotnem 16-bitnem območju. Večina komunikacijskih kanalov sploh ne podpira prenosa paketov, dolgih 2^{16} bajtov, vsaj ne brez fragmentacije. Napadalec prav tako ne more poljubno spremeniti dolžine paketa, saj bi s tem uničil pravilno nezasebno komunikacijo.

3.3.5 Zastavica FIN

V protokol sem vključil tudi kontrolni podatek, ki ponazarja, da je v paketu skrit samo 1 bajt. To se ponavadi zgodi pri pošiljanju lihega števila bajtov zasebnih podatkov. V nasprotnem primeru bi lahko naslovnik zmotno zabeležil oba bajta v zasebno sporočilo, ki bi bilo zato napačno. Za izražanje tega po-

datka sem uporabil zastavico FIN. Zastavica v visokem stanju pomeni, da je v paketu skrit samo 1 bajt. Zastavica ni v protokolu TCP nič nenavadnega in se uporabi ob podiranju povezave, zato napadalec ne more posumiti ničesar. Protokol 1 obenem ne predvideva pošiljanja zasebnih sporočil, ki so krajša od 8 bajtov, kolikor je velikost zasebnih podatkov, ki se pošljejo v prvem paketu.

3.3.6 Deljena skrivnost

Kot vemo, lahko pri steganografiji uporabimo tudi deljeno skrivnost med pošiljateljem in naslovnikom. Ker je varnost pri zagotavljanju zasebnosti ključnega pomena, sem se odločil, da bom v protokol vključil tudi opcijsko uporabo deljene skrivnosti in s tem še malce posegel na področje kriptografije. Kljub temu pa bi o kombinaciji steganografije in kriptografije pri tem protokolu težko govoril, saj sem deljeno skrivnost uporabil za računsko zelo enostaven kriptografski algoritem. Odločil sem se, da bo deljena skrivnost dolga 1 bajt. Za dodatno varnost sem poskrbel tako, da vsak bajt skritega sporočila, ki ga pošljemo po kanalu, dobimo kot rezultat operacije XOR med bajtom iz zasebnega sporočila in deljeno skrivnostjo. Naslovnik izračuna bajt zasebnega sporočila kot rezultat operacije XOR med bajtom skritega sporočila, ki ga prejme po komunikacijskem kanalu, in deljeno skrivnostjo. Prednost operacije XOR je ta, da je računsko zelo enostavna in simetrična. Če pošiljatelj in naslovnik ne želita uporabiti deljene skrivnosti, le-to enostavno postavita na vrednost 0 in zasebno sporočilo se prenese skozi kanal v nekriptirani obliki.

3.4 Protokol 2

Drugi protokol sem zasnoval tako, da dosega višjo kapaciteto nosilca zasebnih podatkov na račun nižje varnosti in robustnosti. Poleg steganografskih tehnik iz protokola 1 sem tukaj uporabil še nekatere druge tehnike, ki jih je mogoče odkriti le z dinamičnimi metodami nadzora.

3.4.1 Sekvenčna številka

V sekvenčno številko sem v tem protokolu skril podatke nekoliko drugače kot pri prvem protokolu. Namesto skrivanja podatkov samo v prvem paketu, sem v tem protokolu uporabil sekvenčno številko za skrivanje podatkov v vseh paketih in tako skril 4 bajte zasebnih podatkov na paket. Zaradi skoraj naključnega spreminjanja sekvenčne številke pa sem izgubil tudi bistvene lastnosti protokola TCP, ki tako ne more več preveriti prejema in vrstnega reda prejetih paketov. Napadalec bi lahko z dinamičnimi metodami nadzora nepravilno zaporedje sekvenčnih številk zaznal, vendar pa zaradi prisotnosti drugih steganografskih tehnik, uporabljenih v tem protokolu, kot bomo videli v nadaljevanju, to ni mogoče.

3.4.2 Identifikacijska številka

Identifikacijska številka se uporablja za sestavljanje fragmentiranih paketov, vendar pa je neničelna vrednost prisotna v vseh paketih. Vrednost naj bi bila po podatkih nekaterih avtorjev pri nefragmentiranih paketih enaka 0, vendar je v realnosti ponavadi naključna ali zaporedna. Zato lahko v identifikacijsko številko skrijemo zasebne podatke in tega s statičnimi metodami ne moremo odkriti. Nepravilnosti lahko odkrijemo le z dinamičnimi metodami, tako da ugotovimo, da identifikacijske številke paketov ne sledijo naključju. S skrivanjem zasebnih podatkov v identifikacijsko številko sem pridobil dodatna 2 bajta skritih podatkov na paket. Napadalec bi sicer, podobno kot pri sekvenčni številki ob inicializaciji povezave, vrednost lahko zamenjal z novo naključno vrednostjo in tako ranil robustnost protokola, vendar pa za to nima pravega razloga.

3.4.3 Izvorni naslov

Zasebne podatke sem pri drugem protokolu skril tudi v izvorni naslov paketa in tako pridobil na kapaciteti nosilca zasebnih podatkov 4 bajte na paket. Ob tem je pomembno, da naslovnik identificira steganografske pakete le s

pomočjo ponornih vrat, saj izvorni naslov oz. naslov pošiljatelja v tem primeru povsem izgubi svoj pomen. Takšno aktivnost je nemogoče zaznati s statičnimi metodami, saj so naslovi IP tudi v običajnih paketih zelo raznoliki in nelogični. Z dinamičnimi metodami pa je aktivnost mogoče zaznati, saj zaporedje paketov k naslovniku ponavadi prihaja z majhnega števila naslovov. Tako je zelo nenavadno, da bi 10 ali več zaporednih paketov dobili s samih različnih naslovov v primeru, da nimamo strežnika s spletno stranjo ali česa podobnega. Spreminjanje izvornega naslova v vsakem paketu pa ima tudi prednost. Navzven je videti, kot da gre pravzaprav za veliko različnih sej TCP, zato sekvenčne številke paketov, ki se skoraj naključno spreminjajo od paketa do paketa, niso sumljive.

3.4.4 Izvorna vrata

Podobno kot izvorni naslov, lahko za steganografijo uporabimo tudi izvorna vrata in v to vrednost skrijemo 2 bajta zasebnih podatkov. Tudi za detekcijo steganografske aktivnosti v izvornih vratih velja enako kot pri izvornem naslovu.

3.4.5 Sekvenčna številka potrditve

Podobno kot v sekvenčno številko, lahko podatke skrijemo tudi v sekvenčno številko potrditve. S tem sicer izgubimo pravo funkcionalnost protokola TCP, vendar pa smo le-to tako ali tako izgubili že s skrivanjem podatkov v preostala polja. Detekcija steganografije v tem polju s statičnimi metodami ni mogoča. Paziti moramo le, da imamo zastavico ACK v visokem stanju. V tem primeru je paket navzven videti povsem normalen. Z dinamičnimi metodami je nepravilnost mogoče zaznati, saj gre za pošiljanje paketa z zastavico ACK brez predhodne inicializacije povezave, vendar pa je takšna detekcija zapletena, saj mora imeti napadalec v pomnilniku vse inicializirane povezave in ves čas opravljati veliko računskega dela.

3.4.6 Urgentni kazalec

Protokol zaenkrat skrije skupaj 18 bajtov zasebnih podatkov znotraj enega paketa. Velika verjetnost je, da pošiljatelj ne bo poslal podatkov, katerih skupna velikost je večkratnik 18-ih bajtov. Zato tudi tu, podobno kot pri prvem protokolu, potrebujemo še kontrolni podatek, ki bo ponazarjal, da paket vsebuje manj kot 18 bajtov zasebnih podatkov in koliko zasebnih podatkov vsebuje. Odločil sem se, da to vrednost skrijem v polje Urgentni kazalec, čeprav za to potrebujem le 5 bitov podatkov, polje pa je dolgo 16 bitov oz. 2 bajta. Vrednost polja različna od 0 ob zastavici URG v visokem stanju ni nič nenavadnega, zato s statičnimi metodami nepravilnosti ni mogoče odkriti. Tudi z dinamičnimi metodami je v tem primeru aktivnost težko izsledljiva. V polje bi bilo mogoče skriti še 1 bajt zasebnih podatkov. S tem bi olajšali detekcijo steganografije z dinamičnimi metodami, kar pa je še vedno v skladu z začetno postavko o varnosti drugega protokola. Kljub temu pa se za to nisem odločil, saj bi ob skrivanju podatkov v zgornji bajt polja vrednost kazalca zlahka presegla skupno dolžino paketa, slednje pa bi bilo izsledljivo zgolj s statičnimi metodami za detekcijo steganografije.

3.5 Protokol 3

Tretji protokol je, kot smo že omenili, najranljivejši protokol iz družine. Pri steganografskih tehnikah tega protokola sem si lahko privoščil, da so le-te izsledljive tudi s statičnimi metodami za detekcijo. Tako sem lahko skril podatke še na nekatera druga mesta zaglavij, kjer v prvih dveh protokolih to ni bilo mogoče. Kljub vsemu pa nas predstava o ranljivosti ne sme zavesti, saj je ta protokol še vedno ostal neopažen s strani Snort-a.

3.5.1 Tip storitve

Tokrat sem se odločil za skrivanje podatkov tudi v polje Tip storitve. Polje je skozi čas nekoliko spreminjalo svoj pomen, njegova funkcija pa je še

danes precej nejasna in ni ravno ključna za protokol IP. Tipična vrednost tega polja je 0, v navadnih paketih pa so možne tudi nekatere druge vrednosti. Naključne vrednosti tega polja so sumljive, zato je mogoče nepravilnosti detektirati že z uporabo statičnih metod.

3.5.2 Odmik fragmenta

Ob zastavici Ne fragmentiraj v visokem stanju vrednost polja Odmik fragmenta izgubi svoj pomen. Pri navadnih paketih je v takšnem primeru vrednost tega polja postavljena na 0, zato jo lahko nadomestimo s 13 biti iz zasebnega sporočila. Takšno početje je sicer izsledljivo zgolj s statičnimi metodami napadalca, vendar si to v skladu z načeli tretjega protokola lahko privoščimo. Težave pri takšnem početju so se pojavile šele ob implementaciji, saj takšni paketi niso prispeli do naslovnika. Z iskanjem mesta izgube paketa se nisem ukvarjal, temveč sem to steganografsko tehniko izpustil iz protokola.

3.5.3 Zastavice TCP

Glava protokola TCP vsebuje 1 bajt veliko polje za zastavice, v katerih so zelo pomembni kontrolni podatki. Funkcionalnosti protokola TCP pa sem se odrekel že pri drugem izmed družine protokolov, zato sem lahko tudi vrednost osmih zastavic zamenjal z 1 bajtom zasebnega sporočila. Žal pa protokol TCP ne dopušča vseh 256 možnosti postavitve zastavic. Skrajni primer je kombinacija, kjer so vse zastavice v visokem stanju in zagotovo ni dovoljena znotraj TCP protokola. Tako lahko napadalec samo s pomočjo statičnih metod pregleda, ali je kombinacija zastavic dovoljena, v nasprotnem primeru pa posumi na uporabo steganografije.

3.5.4 Urgentni kazalec

Polje Urgentni kazalec sem sicer za steganografijo uporabil že v drugem protokolu, vendar je šlo tam za drugačno uporabo. V tretjem protokolu sem vre-

dnost tega polja v celoti nadomestil z 2 bajtoma zasebnih podatkov. Takšno početje seveda ni odporno na statične metode za detekcijo, saj lahko pride tako do neujemanja skupne dolžine paketa in vrednosti kazalca kot tudi do neujemanja zastavice URG v nizkem stanju in neničelne vrednosti kazalca.

3.5.5 Rezervirano in Ne fragmentiraj

Tretji protokol skrije 23 bajtov zasebnih podatkov znotraj enega paketa. Tudi tu je treba zagotoviti kontrolne podatke za primer, ko se v paketu pošlje manj kot 23 bajtov zasebnih podatkov. Za ponazoritev števila bajtov zasebnega sporočila v paketu potrebujemo 5 bitov. Spodnje 4 bite podatkov sem skril v rezervirano polje glave TCP. Polje je rezervirano za poznejšo uporabo in ima pri navadnih paketih vrednost 0, zato neničelna vrednost nakazuje na steganografijo, kar napadalec lahko detektira s statičnimi metodami. Zgornji bit sem skril kar v bit Ne fragmentiraj, saj vrednost bita ne vpliva na komunikacijo, če fragmentacija ni potrebna.

Poglavje 4

Implementacija protokolov

Implementacijo sem pravzaprav opravljal vzporedno z zasnovo protokolov, saj sem tako lahko sproti testiral, ali steganografske tehnike v danem okolju zares delujejo in kako težko jih je detektirati. Še prej pa sem moral izbrati implementacijsko okolje.

4.1 Implementacijsko okolje

Pred izbiro implementacijskega okolja sem veliko razmišljal o praktični uporabi svojih protokolov. Prišel sem do zaključka, da bi bili protokoli uporabni predvsem pri zasebnem pošiljanju krajših sporočil in na napravah, kjer je procesorska moč zelo omejena. Implementacijo sem kljub temu izvedel na namiznem računalniku, saj je takšna implementacija najlažje izvedljiva. Kodo sem napisal v programskem jeziku C, saj ima ta dobro podporo za manipuliranje podatkov na nizkem nivoju. Spreminjanje glave TCP, predvsem pa IP je namreč precej nizkonivojsko opravilo, saj s temi podatki ponavadi upravlja operacijski sistem in ne visokonivojski programer. Večina knjižnjic jezika C je kompatibilna z operacijskim sistemom Linux in to je tudi razlog, da sem le-tega izbral za implementacijo.

4.2 Izbira zasebnih in nezasebnih podatkov

Steganografske tehnike so v večini primerov odvisne od tipa nezasebnih podatkov. Če na primer skrivamo podatke v tekst, se tehnike povsem razlikujejo od tistih, ki so namenjene skrivanju v sliko. Pri mojih protokolih so uporabljene tehnike, ki podatke skrivajo v glave protokolov TCP in IP. Tako tip nezasebnih podatkov, ki jih pošiljamo v vsebini paketa, nima vidnega vpliva na samo steganografijo. Tehnike steganografije so odvisne zgolj od kontrolnih podatkov komunikacije. To pomeni, da lahko pri teh protokolih pošiljamo poljubne nezasebne podatke.

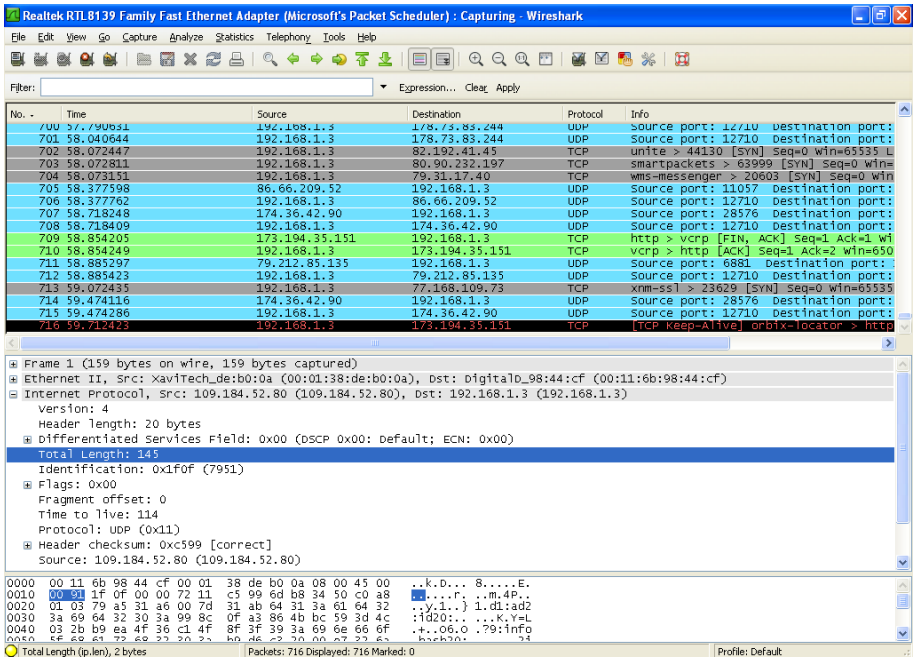
Podobno velja tudi za tip zasebnih podatkov. Ne glede na tip moramo podatke vedno zakodirati v zaporedje bitov. Protokol množico bitov, ki spadajo med zasebne podatke, skrije v paket. Kodiranje in format zasebnih podatkov sta povsem nepomembna, saj protokol deluje na nivoju bajta in se z njegovo vsebino ter pomenom ne ukvarja.

Implementacija deluje tako, da uporabnik, ki želi poslati zasebne podatke, poda programu zasebno in nezasebno datoteko. Na drugi strani sprejemnik določi ime obeh datotek, v katere se shranijo prejeti podatki. Format datotek je poljuben in odvisen od končnice datoteke.

4.3 Wireshark

Ob implementaciji protokola je bilo zame zelo pomembno, da lahko spremljam promet, ki teče po komunikacijskem kanalu, saj ta promet lahko spremlja tudi napadalec. Poleg tega mi je pogled v promet tudi močno olajšal iskanje napak v kodi, saj je napake v delovanju zelo težko locirati, če nimamo vpogleda v prenosni medij. Ko vidimo, kaj se po mediju dejansko prenaša, lahko hitro ugotovimo vsaj to, ali je napaka pri pošiljatelju ali pri naslovniku.

V ta namen bi lahko uporabil že omenjeni program Snort v načinu Sniffer, vendar sem se zaradi boljše preglednosti odločil za program Wireshark. Gre za odličen program, ki ponuja tudi grafično bogat uporabniški vmesnik in tako omogoča enostavno spremljanje paketov, ki potujejo po kanalu. Upo-



Slika 4.1: Wireshark v grafičnem načinu

rabniški vmesnik prikazuje Slika 4.1. Poleg tega nam program omogoča še veliko drugih možnosti. Tako lahko na primer pakete poljubno filtriramo.

Program omogoča tudi enostaven vpogled v glave različnih protokolov, saj sam označi, kateri podatek v paketu pripada kateremu polju in za to uporabniku ni treba skrbeti. V nasprotnem primeru bi moral uporabnik na pamet poznati strukture glav. Prav tako Wireshark analizira tudi kontrolne podatke v glavah in uporabniku prikaže smiselno razlago o tipu paketa.

4.4 Podrobnosti implementacije

V tem podpoglavju bom na kratko povzel podrobnosti implementacije in v opis vključil nekaj pomembnejših delov kode. Pri implementaciji protokolov sem si pomagal tudi z implementacijo C. H. Rowlanda [6], v kateri je implementiral nekaj steganografskih tehnik za vdor v strežnik.

Implementacija je nekoliko poenostavljena in ne upošteva vseh lastnosti

TCP protokola. Ta je namreč že sam posebi zelo zapleten in zahteva redno pošiljanje potrditev o sekvenčnih številkah prejetih paketov in še veliko drugih stvari. Sam sem zaradi nezahtevnosti, tudi pri implementaciji prvega, najbolj varnega protokola, okrnil protokol TCP do te mere, da protokol nudi funkcionalnost protokola UDP, čeprav gre fizično za paket TCP. Pošiljatelj tako pošilja pakete enega za drugim kljub temu, da naslovnik prejema paketov ne potrdi, saj manjka implementacija logike protokola TCP. Implementacija vseh funkcionalnosti protokola TCP bi bila precej kompleksna in ni namen tega dela.

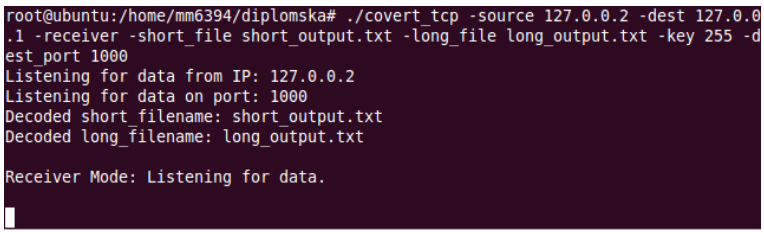
4.4.1 Uporaba programa

Uporabnik program zažene z Linuxovega terminala in ob tem poda ustrezne parametre, ki so pri prvem protokolu:

- IP naslov pošiljatelja,
- IP naslov naslovnika,
- št. vrat naslovnika,
- stikalo pošiljatelj/naslovnik,
- ime zasebne datoteke,
- ime nezasebne datoteke,
- deljena skrivnost.

Program na začetku preveri pravilno rabo parametrov in v primeru napake izpiše ustrezen odziv. Deljena skrivnost ni obvezen parameter. Če ga uporabnik ne uporabi, program za deljeno skrivnost uporabi vrednost 0.

Stikalo pošiljatelj/naslovnik deluje tako, da v primeru, da želimo biti naslovnik, ukazu dodamo parameter *-receiver*. V nasprotnem primeru bo program deloval v načinu pošiljatelja. Vsi ostali parametri so obvezni.



Slika 4.2: Klic programa pri naslovniku

4.4.2 Testni podatki

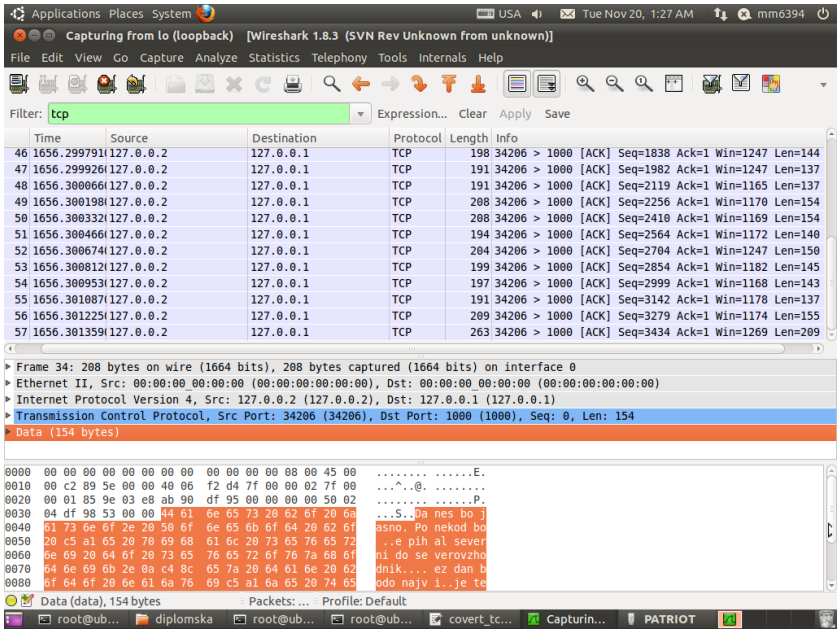
Slika 4.2 prikazuje primer klica programa na naslovnikovem računalniku. Implementacijo sem v tem primeru testiral na enem samem računalniku, zato sem uporabljal naslove IP 127.0.0.1 in 127.0.0.2, ki pomenijo naslov lastnega računalnika. Takoj po klicu programa se na standardni izhod izpiše nekaj osnovnih podatkov. Ko računalnik začne prejemati zasebne podatke, pa se tudi ti izpisujejo na standardni izhod v ASCII vrednostih, vzporedno pa se zapisujejo tudi v izhodno datoteko.

Programa preostalih dveh protokolov sta podobna, vendar zahtevata nekaj manj parametrov. Tako na primer izvorni naslov ni potreben, saj se vanj pozneje zapišejo zasebni podatki.

Programe sem preizkušal na več različnih zasebnih in nezasebnih datotekah. Največ časa sem uporabljal tekstovne datoteke, saj je z njimi razhroščevanje najlažje. Slika 4.3 prikazuje analizo prometa z uporabo prvega protokola, kjer sem za zasebne podatke uporabil tekstovno datoteko z vsebino *To je zasebno sporočilo* in za nezasebne podatke tekstovno datoteko z vremensko napovedjo. V spodnjem delu slike vidimo analizo prvega poslanega paketa. Opazimo lahko, da je v dejanski vsebini paketa (označeni del podatkov) le vremenska napoved.

4.4.3 Uporabljene knjižnice

V kodi sem uporabil kar nekaj knjižnic. Nekatere izmed njih so zelo splošne, večina pa jih je nekoliko nevsakdanjih in vsebujejo predvsem funkcije in



Slika 4.3: Primer poslanega zasebnega sporočila

strukte za delo z računalniškimi komunikacijami. Uporabil sem naslednje knjižnjice:

- stdio,
- stdlib,
- signal,
- string,
- unistd,
- netdb,
- netinet/in,
- sys/socket,
- arpa/inet,

- linux/ip,
- linux/tcp.

Knjižnice **stdio**, **stdlib** in **string** so zelo splošne. Uporabil sem jih za upravljanje s pomnilnikom, manipulacijo nizov, pretvorbe, vhodno–izhodne operacije in podobno.

Preostale knjižnice so namenjene predvsem podpori za računalniške komunikacije, nekatere specifično za operacijski sistem Linux. Omogočile so mi enostavnejšo vzpostavitev povezave med pošiljateljem in naslovnikom. Med drugim omogočajo tudi enostaven vnos podatkov v posamezna polja glav TCP in IP.

4.4.4 Izvorna koda

V tej sekciji bom na kratko predstavil pomembnejše dele izvorne kode za boljšo predstavo o dejanski implementaciji v programskem jeziku C.

Koda 4.1 prikazuje vstavljanje lastnih podatkov v vsa polja glav protokolov TCP in IP pri prvem paketu. Koda pripada prvemu izmed treh protokolov za zagotavljanje zasebnosti, zato je večina vrednosti privzetih, z izjemo štirih polj, v katera skrijemo zasebne podatke. Prvih 8 bajtov zasebnih podatkov je shranjenih v tabeli *ch*. Ti podatki so na tem mestu tudi že zakriptirani z deljeno skrivnostjo. Prve 4 bajte podatkov skrijemo v **sekvenčno številko (seq)**, 5. bajt pa v spodnjih 8 bitov polja **Skupna dolžina (tot_len)**. Polju Skupna dolžina nato prištejemo še dolžino glave IP. Šesti bajt shranimo v spodnjih 8 bitov polja **Okno (Window)**, vendar vrednosti prištejemo konstanto 1024, saj so vrednosti polja nad 1024 bolj običajne. Zadnja 2 bajta zasebnih podatkov skrijemo v **izvorna vrata(source)**. Funkciji *htons()* in *htonl()* sta uporabljeni za zamenjavo tankega in debelega konca podatkov (16/32 bit), saj mrežna kartica uporablja drugačno ureditev, kot pa programski jezik C. Vrednost kontrolnih vsot je zaenkrat postavljena na 0, vendar se pozneje v programu le-ta nadomesti z ustrezno izračunano vrednostjo. Koda za nadaljnje pakete je malenkost drugačna v smislu postavitve zastavic in

sekvenčne številke.

```
send_tcp.ip.ihl = 5;
send_tcp.ip.version = 4;
send_tcp.ip.tos = 0;
len = 20 + (ch[4]);
send_tcp.ip.tot_len = htons(len);
send_tcp.ip.id = rand();
send_tcp.ip.frag_off = 0;
send_tcp.ip.ttl = 64;
send_tcp.ip.protocol = IPPROTO_TCP;
send_tcp.ip.check = 0;
send_tcp.ip.saddr = source_addr;
send_tcp.ip.daddr = dest_addr;
src_port = 256*ch[6] + ch[7];
send_tcp.tcp.source = htons(src_port);
seq = 16777216*ch[0] + 65536*ch[1] + 256*ch[2] + ch[3];
send_tcp.tcp.seq = htonl(seq);
send_tcp.tcp.dest = htons(dest_port);
send_tcp.tcp.ack_seq = 0;
send_tcp.tcp.res1 = 0;
send_tcp.tcp.doff = 5;
send_tcp.tcp.fin = 0;
send_tcp.tcp.syn = 1;
send_tcp.tcp.rst = 0;
send_tcp.tcp.psh = 0;
send_tcp.tcp.ack = 0;
send_tcp.tcp.urg = 0;
send_tcp.tcp.cwr = 0;
send_tcp.tcp.ece = 0;
send_tcp.tcp.window = htons(ch[5] + 1024);
send_tcp.tcp.check = 0;
```

```
send_tcp.tcp.urg_ptr = 0;
```

Koda 4.1: Vstavljanje podatkov v glavi TCP in IP

Koda 4.2 je zaslužna za pošiljanje paketa naslovniku. Gre za zaporedje vrstic, ki najprej inicializirajo vtič, na koncu pa se v funkciji *sendto()* pošlje paket. Funkciji podamo vtič, kazalec na podatke, ki jih želimo poslati, dolžino podatkov, zastavice, kazalec na *struct sockaddr* in dolžino tega strukta.

```
sin.sin_family = AF_INET;
sin.sin_port = send_tcp.tcp.source;
sin.sin_addr.s_addr = send_tcp.ip.daddr;
send_socket = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
sendto(send_socket, &send_tcp, len, 0,
(struct sockaddr *)&sin, sizeof(sin));
```

Koda 4.2: Pošiljanje paketa

Koda 4.3 pa je namenjena naslovniku. Ta sprva zazna prejeti paket in ga shrani v pomnilnik. Nato pregleda, ali sta izvorni naslov in ponorna vrata paketa res namenjena zasebni komunikaciji. Ta del je seveda v preostalih dveh protokolih drugačen, saj mora tam naslovnik pregledati vse pakete, ki so naslovljeni na vnaprej določena vrata. Preverjanje zastavice *SYN*, postavljene na 1, je prisotno zgolj zato, ker je del kode namenjen prvemu paketu iz zasebne komunikacije. Za nadaljnje pakete je koda nekoliko drugačna, saj tam prejmemo le 2 bajta zasebnih podatkov na paket. Program nato pridobi podatke iz ustreznih polj glav TCP in IP in jih zapiše v datoteko, vzporedno pa jih izpiše tudi na standardni izhod.

```
recv_socket = socket(AF_INET, SOCK_RAW, 6);
read(recv_socket, (struct recv_tcp *)&recv_pkt, 9999);
/* Ce je pravi izvorni naslov in ponorni port */
if((recv_pkt.ip.saddr == source_addr) &&
(htons(recv_pkt.tcp.dest) == dest_port))
{
```

```
/* Preberi zasebno sporocilo */
if(recv_pkt.tcp.syn == 1)
{
ch[0] = ((htonl(recv_pkt.tcp.seq) / 16777216) & 255) ^
key;
ch[1] = ((htonl(recv_pkt.tcp.seq) / 65536) & 255) ^
key;
ch[2] = ((htonl(recv_pkt.tcp.seq) / 256) & 255) ^
key;
ch[3] = (htonl(recv_pkt.tcp.seq) & 255) ^ key;
ch[4] = ((htons(recv_pkt.ip.tot_len) & 255) - 20) ^
key;
ch[5] = (htons(recv_pkt.tcp.window) & 255) ^ key;
ch[6] = ((htons(recv_pkt.tcp.source) / 256) & 255) ^
key;
ch[7] = (htons(recv_pkt.tcp.source) & 255) ^ key;
printf("Receiving Data: %c%c%c%c%c%c%c%c\n", ch[0],
ch[1],ch[2],ch[3],ch[4],ch[5],ch[6],ch[7]);
fprintf(output,"%c%c%c%c%c%c%c%c", ch[0],ch[1],ch[2],
ch[3],ch[4],ch[5],ch[6],ch[7]);
fflush(output);
}
```

Koda 4.3: Prejemanje paketa

Poglavje 5

Evalvacija protokolov

V tem poglavju bom na kratko evalviral družino protokolov s stališča štirih glavnih parametrov, ki smo jih že spoznali.

5.1 Protokol 1

5.1.1 Kapaciteta

Kapaciteta nosilca zasebnih podatkov nam pove, koliko zasebnih podatkov lahko pošljemo znotraj nezasebnih. Z uporabo tega protokola lahko v prvem paketu nezasebnih podatkov pošljemo vzporedno še 8 bajtov, v vseh nadaljnjih paketih pa 2 bajta zasebnih podatkov. Razlog za to je ta, da v polji Sequence number in Source port skrijemo podatke le v prvem paketu.

Protokol ima dodatno zanimivost. Skupna velikost paketa se namreč spreminja glede na zasebne podatke, saj 1 bajt le-teh skrijemo tudi v polje Skupna dolžina glave IP. To pomeni, da natančnega deleža zasebnih podatkov v nezasebnih ne moremo določiti. To je v smislu evalvacije morda malce moteče, saj nas bolj zanima delež zasebnih podatkov v paketu, kot pa njihova absolutna velikost. Nezasebni podatki namreč prinašajo dodaten strošek in zakasnitev, čemur se v protokolu želimo izogniti. Na žalost takšna evalvacija v tem protokolu ni mogoča, zato se moramo zadovoljiti s podatkom kapacitete osmih oz. dveh bajtov zasebnih podatkov znotraj enega paketa.

5.1.2 Varnost

Varnost je pri zagotavljanju zasebnosti zagotovo ključnega pomena. Pri uporabi steganografije ponavadi celotna varnost sloni na predpostavki, da napadalec pri paketih ne bo opazil nepravilnosti.

Komunikacija pri tem protokolu je sicer zelo podobna običajni, nesteganografski komunikaciji, vendar pa so kljub temu opazne nekatere razlike. Problematično je namreč spreminjanje dolžine paketa in vrednosti polja Okno. Spreminjanje teh vrednosti je sicer običajno in nesumljivo, vendar se običajno te vrednosti ne spremenijo v vsakem naslednjem paketu. Ko v ta polja vstavljamo zasebne podatke, pa je verjetnost za spremembo vrednosti v vsakem naslednjem paketu zelo velika.

Podobno so pri običajni komunikaciji tudi spremembe vrednosti nekoliko drugačne. Gre za spreminjanje le med nekaj različnimi vrednostmi, ki se pogosto pojavijo. V nasprotju s tem je pri komunikaciji po mojem protokolu 256 vrednosti, ki se na dolgi rok pojavljajo enako pogosto. Problem varnosti je torej predvsem statistične narave. Predpostavka, da protokol uporabimo v okolju, kjer je velikost zasebnih podatkov majhna, nam pride na področju varnosti zelo prav, saj je statistična detekcija sumljivega dogajanja s tem otežena.

A vendarle ima protokol s stališča varnosti tudi precej dobrih lastnosti. Poskrbel sem namreč, da je zaloga vrednosti obeh polj omejena na vrednosti znotraj enega bajta in tako precej zmanjšal sumljivo razpršenost vrednosti. Prav tako je spreminjanje teh dveh vrednosti vsakdanje in ni nekaj, kar splošno nakazuje na sumljivo dejanje, kot je to značilno za nekatere druge steganografske tehnike. Prav to dejstvo pa močno otežuje avtomatizirano detekcijo steganografije, ki je najbolj pogosta in najnevarnejša vrsta detekcije. Dinamične metode, ki bi takšno steganografijo znale odkriti, bi morale biti precej zapletene. Ob tem pa bi napadalec najverjetneje dobil veliko lažnih opozoril o sicer povsem pravilnih paketih iz vsakdanje komunikacije.

Protokol ima zaradi ključnega pomena varnosti tudi možnost uporabe deljene skrivnosti. Ta zagotovo nekoliko izboljša varnost, saj tudi v primeru,

da napadalec posumi na skrivanje podatkov v omenjenih poljih, ob njihovem preverjanju ne bo prišel do smiselne interpretacije le-teh in bo zato najverjetneje ovrgel sum na prikrito, zasebno komunikacijo.

5.1.3 Robustnost

Mera robustnosti nam pove, kolikšna je verjetnost, da bodo zasebni podatki dosegli naslovnika nespremenjeni. Protokol je s stališča robustnosti zasnovan zelo dobro. Robustnost je najbrž prvina, ki je pri tem protokolu najmočnejša.

Največja pomanjkljivost robustnosti tega protokola je v polju Sekvenčna številka prvega paketa. Čeprav bi bilo to nenavadno, lahko napadalec vrednost spremeni in s tem ne vpliva na nezasebno komunikacijo. To pomeni, da se bodo 4 bajti zasebnih podatkov do naslovnika prenesli napačno.

Po drugi strani pa spremembe drugje niso možne. Vsaka sprememba v kateremkoli izmed preostalih treh polj, ki se uporabljajo za steganografijo, bi pomenila tudi viden vpliv na nezasebno komunikacijo. To bi sicer vseeno pomenilo, da naslovnik ne bi prejel pravilnih podatkov, vendar pa na splošno ne moremo vplivati na napadalca v tej meri, da mu preprečimo spreminjanje podatkov v paketu. Znotraj protokola lahko dosežemo le to, da bo napadalec s spremembo zasebnih podatkov vplival tudi na nezasebno komunikacijo.

Protokol je odporen tudi na druge napake v komunikaciji. Za to, da ne pride do napak na nivoju posameznega bita, skrbi nižjeležeči protokol na povezavni plasti z uporabo kontrolnih vsot. Za to, da se paket ne izgubi, pa skrbi kar sam protokol TCP z rednim pošiljanjem potrditev prejetih sekvenčnih številk paketov.

5.1.4 Računska nezahtevnost

Protokol, ki sem ga zasnoval, je kljub precej dolgi kodi matematično zelo enostaven. Gre namreč, kot pri večini steganografskih tehnik, zgolj za spreminjanje podatkov na točno določenih mestih. Če izvzamemo branje in pisanje datotek, je delo računalnika le razbijanje podatkov na nivo bajta in

vstavljanje le-teh na ustrezna mesta v glavah TCP in IP. Pri sprejemniku je zgodba obratna, vendar računsko enako enostavna.

Računska zahtevnost bi se lahko skrivala tudi v kriptografskem delu protokola, kjer podatke zakriptiramo z uporabo deljene skrivnosti. Vendar pa kriptiranje deluje zgolj na principu operacije XOR med dvema znakoma. Računalniški procesorji imajo mehanizem za računanje te operacije vključen že strojno v aritmetično-logični enoti, zato je operacija na današnjih računalnikih računsko nezahtevna.

5.2 Protokol 2

Ta protokol uporablja steganografske tehnike prvega protokola in še nekaj dodatnih, zato zanj veljajo nekatere evalvacijske ocene prvega protokola. Kljub temu pa so zaradi novih tehnik prisotne pomembne spremembe, ki zahtevajo ponovno evalvacijo.

5.2.1 Kapaciteta

Protokol v primerjavi s prvim precej blesti po preskoku v kapaciteti. Namesto dveh bajtov na paket, lahko s tem protokolom pošljemo kar 18 bajtov zasebnih podatkov znotraj enega samega paketa. To pomeni, da lahko z zamenjavo protokola povečamo kapaciteto nosilca zasebnih podatkov za kar 9-krat. To nam lahko pride prav predvsem pri senzorskih omrežjih, saj tako pošljemo zasebno sporočilo v manj paketih in s tem podaljšamo življensko dobo naprave, ki je napajana na bateriji.

Še vedno pa je tisto pravo kapaciteto težko oceniti, saj zaradi nekonstantne skupne dolžine paketa, enako kot pri prvem protokolu, tudi tu ne moremo točno napovedati deleža zasebnih podatkov znotraj nezasebnih.

5.2.2 Varnost

Varnost je pri tem protokolu na nekoliko nižjem nivoju, vendar za večino okoliščin še vedno sprejemljiva. Ranljivost se skriva v dinamičnih metodah za detekcijo steganografije, ki sem jih poizkušal implementirati tudi sam.

Statistična nepravilnost, ki jo je mogoče detektirati z dinamičnimi metodami, se sicer pojavi že pri prvem protokolu. Vendar pa se pri drugem protokolu statistične nepravilnosti zelo nakopičijo. Največji problem predstavlja konstantno spreminjanje izvornega naslova in izvornih vrat. Pri pošiljanju majhnega števila paketov je tveganje neznatno, vendar le-to močno naraste z večanjem števila poslanih paketov, saj je raznolikost izvornega naslova in vrat v sosednjih paketih z večanjem števila paketov vedno bolj nenavadna. Pri samo nekaj poslanih paketih to ni nič nenavadnega in se dogaja tudi pri povsem normalni komunikaciji.

Podobno velja tudi za vrednost identifikacijske številke, vendar je tu zadeva obratna. Problem pri identifikacijski številki se pojavi zaradi premajhne raznolikosti v primeru steganografije. Detekcija takšnega početja pa je zaradi naravne raznolikosti zasebnih podatkov mnogo bolj zahtevna in zahteva dobro premišljen statistični pristop.

5.2.3 Robustnost

V primerjavi s prvim protokolom pri tem protokolu nekoliko pade tudi mera robustnosti. To pa se večinoma ne zgodi zaradi napadalca, razlog se skriva drugje.

Protokol namreč odvzame funkcionalnost protokolu TCP in zato nimamo več preverjanja prejetih paketov in njihovega vrstnega reda. Zato se lahko zaradi napake v komunikacijskem kanalu paketi izgubijo ali pa zamenjajo vrstni red, naslovnik pa tega ne bo mogel detektirati.

Prav tako pa lahko tudi tu napadalec enostavno zamenja kontrolne podatke. Izvornega naslova in izvornih vrat ne sme spreminjati, saj bi s tem lahko vplival na nezasebno komunikacijo. Kljub temu pa lahko spremeni

vrednost identifikacijske številke in s tem ohrani lastnosti nezasebne komunikacije, kar dodatno zmanjšuje mero robustnosti protokola.

5.2.4 Računska nezahtevnost

Računska nezahtevnost je, podobno kot pri prvem protokolu, tudi pri tem zelo visoka. Nezahtevnost se za spoznanje zmanjša le zaradi večjega števila zasebnih podatkov, ki jih je treba skriti v kontrolne podatke paketov. Vendar pa gre tudi tu zgolj za zamenjavo podatkov, kar je računsko zelo enostavna, zato bistvenega padca v računski nezahtevnosti pri tem protokolu v primerjavi s prvim ni zaznati.

5.3 Protokol 3

Protokol prav tako vsebuje nekatere značilnosti, ki so že omenjene pri prvih dveh protokolih, vendar je treba zaradi dodanih steganografskih tehnik tudi ta protokol ponovno evalvirati.

5.3.1 Kapaciteta

Tretji protokol je protokol z najvišjo kapaciteto nosilca zasebnih podatkov in pošilja kar 23 bajtov zasebnih podatkov znotraj enega paketa, kar znaša dobro polovico vseh kontrolnih podatkov (40 bajtov).

Tudi tu točnega deleža zasebnih podatkov znotraj nezasebnih ne moremo določiti. Protokol je zasnovan tako, da lahko skupna dolžina paketa znaša največ 255 + 40 bajtov, kar pomeni, da v najslabšem primeru paket vsebuje skoraj 8 odstotkov zasebnih podatkov. Delež je precej impresiven, če zopet ponovim dejstvo, da Snort ni opazil nepravilnosti v paketih tega protokola. Ob današnjih hitrostih računalniških komunikacij bi tako lahko z uporabo tega protokola poslali celotno diplomsko delo v nekaj sekundah.

5.3.2 Varnost

Varnost je pri tem protokolu postavljena nekoliko na stran. Razlog je v skrivanju zasebnih podatkov v polja, kot sta Rezervirano in Zastavice. Polje Rezervirano z vrednostjo, različno od nič, je tipičen in zelo šolski primer uporabe steganografije v računalniških komunikacijah, zato napadalcu ni treba biti zelo podkovan na tem področju, da odkrije nepravilnosti v paketih.

Varnost tega protokola temelji predvsem na predpostavki, da napadalec enostavno ne bo ročno nadzoroval kontrolnih podatkov, ampak bo poizkušal zaznati nepravilnosti v paketih s pomočjo Snort-a ali podobnega programa, ki je pretežno specializiran za odkrivanje nepravilnosti v sami vsebini paketa.

5.3.3 Robustnost

Robustnost je pri tem protokolu še vedno na podobnem nivoju kot pri drugem. Napadalec namreč ne more spremeniti vrednosti polj Tip storitve, Urgentni kazalec in Zastavice, saj bi s tem vplival tudi na nezasebno komunikacijo. Lahko pa spremeni vrednost polja Rezervirano, vendar s tem vpliva zgolj na kontrolni podatek o številu bajtov zasebnih podatkov v paketu. Robustnost zato ostaja na podobnem nivoju, kot pri drugem protokolu.

5.3.4 Računska nezahtevnost

Podobno kot pri drugem se računska nezahtevnost tudi pri tem protokolu bistveno ne spremeni in se morda le za spoznanje zmanjša zaradi dodatnih štirih bajtov, ki jih je treba skriti v paket.

Poglavje 6

Sklepne ugotovitve

V tej diplomski nalogi sem zasnoval družino protokolov za zagotavljanje zasebnosti in le-to implementiral. Protokoli so morali zasebnost zagotavljati na podlagi steganografije in ne kriptografije, ki je danes bolj pogosta. Implementacija je bila narejena za računalniške komunikacije po omrežnem modelu TCP/IP.

Ugotovil sem, da je na področju steganografije v računalniških komunikacijah razvitih zelo veliko tehnik. Ko sem začel brskati po literaturi, sem pričakoval le nekaj uporabnih rešitev, našel pa sem ogromno pristopov. Omejil sem se na pristope znotraj protokolov TCP in IP, čeprav bi bilo mogoče družino protokolov razširiti tudi na večino protokolov preostalih plasti modela TCP/IP. TCP in IP sta vendarle protokola, ki imata veliko funkcionalnosti in zato veliko možnosti za steganografijo.

Pri pregledovanju pristopov iz literature sem opazil, da so namenjeni predvsem eksperimentalni naravi ali pa vdorom v strežnik. Posledica tega je, da je varnost tam nekoliko manjšega pomena kot pri protokolu za zagotavljanje zasebnosti. Težava se pojavi predvsem takrat, ko ima napadalec enako ali boljše znanje o steganografiji kot tisti, ki je zasnoval protokol. To je bil glavni vzrok tega, da sem skušal najti pristope, ki bi kljub dobremu znanju napadalca le-temu še vedno otežili dostop do zasebnih podatkov. To sem dosegel z nekoliko preprostejšimi steganografskimi tehnikami, ki kljub

temu dajejo zelo dobre rezultate na vseh področjih. Zaradi podobnega razloga sem v protokole dodal tudi možnost uporabe deljene skrivnosti in tako v družino protokolov vključil še enostavno kriptografijo.

Z implementacijo protokolov večjih težav ni bilo, zagotovo pa mi je bila v pomoč tudi implementacija C. H. Rowlanda [6]. Kljub temu sem za to potreboval kar precej časa, saj so napake pri programiranju glav omrežnih protokolov pogoste in jih je razmeroma težko odkriti. V veliko pomoč mi je bil program Wireshark, saj sem z njim lahko spremljal dejansko vsebino paketov, ki se pošiljajo po omrežju in tako veliko lažje razumel napake, ki so se pojavljale pri prenosu paketov. Precej težav sem imel tudi s konfiguracijo programa Snort.

Prvi protokol je namenjen okoliščinam, kjer sta varnost in robustnost na prvem mestu. Protokol je zelo enostaven, zato ga je zagotovo mogoče izboljšati ali razširiti. Predvsem je treba razmisliti o okoliščinah, v katerih se protokol uporablja, saj so v drugačnih okoliščinah lahko drugačni pristopi veliko bolj uspešni. Protokol sem zasnoval na predpostavki, da bo deloval za pošiljanje majhne velikosti zasebnih podatkov in na razmeroma preprostih napravah, kjer je varnost še vedno ključnega pomena. Po mojem mnenju se protokol v takšnih okoliščinah obnese zadovoljivo, predvsem pa je zelo enostaven za razumevanje in implementacijo. Skrivanje podatkov v dolžino paketa lahko uporabimo v vsakem protokolu s spremenljivo dolžino podatkov.

Protokol bi bilo brez sprememb možno implementirati tudi znotraj IP verzije 6, saj večina steganografskih tehnik sloni na protokolu TCP, ki pa je enak ne glede na verzijo protokola IP. Razlika pri implementaciji za protokol IP verzije 6 je ta, da protokol nima več polja Skupna dolžina, ampak ima namesto njega polje Dolžina koristne vsebine (Payload length). Polje je prav tako 16-bitno, od polja Skupna dolžina verzije 4 pa se razlikuje v tem, da v vrednost ni več vključena dolžina glave protokola, temveč le dolžina dejanskih podatkov, ki jih prenesemo v paketu in sledijo glavi.

Drugi protokol dosega kar 9-krat višjo kapaciteto nosilca zasebnih podatkov na račun nekoliko slabše varnosti in robustnosti. Steganografijo je

znotraj protokola mogoče detektirati le z dinamičnimi metodami nadzora, ki zahtevajo shranjevanje velikega števila paketov v pomnilnik in zahtevne statistične preglede nad njimi. Protokol je varen in primeren predvsem za kratka sporočila, ki zaradi majhnega števila poslanih paketov otežujejo statistično detekcijo nepravilnosti.

Protokol v primerjavi s prvim uporablja veliko več steganografskih pristopov in je zato bolj specifičen ter težje prenosljiv na komunikacije, ki ne delujejo po modelu TCP/IP, prav tako pa bi bile potrebne večje spremembe protokola za uporabo znotraj protokola IP verzije 6.

S tretjim protokolom sem poizkušal doseči kar maksimalno kapaciteto zasebnih podatkov. Protokol sicer dosega približno 30 odstotkov višjo kapaciteto zasebnih podatkov od drugega, vendar pa je tudi bistveno bolj ranljiv, saj postavlja vrednosti kontrolnih podatkov na kombinacije, ki so v normalnih okoliščinah nesmiselne. Protokol je uporaben predvsem v okoliščinah, kjer sta optimalnost delovanja in kapaciteta nosilca zasebnih podatkov v paketu pomembni, obenem pa dobro poznamo zmožnosti napadalca.

Literatura

[1] (2001) IP Checksum Covert Channels and Selected Hash Collision. Dostopno na:
http://gray-world.net/papers/ipccc.pdf

[2] (2002) Covert Channel Analysis and Data Hiding in TCP/IP. Dostopno na:
http://gray-world.net/papers/ahsan02.pdf

[3] W. Bender, W. Butera, D. Gruhl, R. Hwang, F. J. Paiz, S. Pogreb, "Techniques for data hiding," *IBM SYSTEMS JOURNAL*, št. 35, zv. 3–4, str. 313–336, 1996.

[4] S. Cabuk, C. E. Brodley, C. Shields, "IP Covert Timing Channels: Design and Detection," v zborniku *CCS '04: Proceedings of the 11th ACM Conference on Computer and Communications Security*, New York, USA, oct. 2004, str. 178–187.

[5] (2002) Covert Messaging Through TCP Timestamps. Dostopno na:
http://web.mit.edu/ greenie/Public/asrg.pdf

[6] (1997) Covert Channels in the TCP/IP Protocol Suite. Dostopno na:
http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/528/449

[7] (2011) Sistem za omrežno detekcijo vdora – Snort. Dostopno na:
https://ucilnica1112.fri.uni-lj.si/mod/resource/view.php?id=12273

[8] S. Zander, G. Armitage, P. Branch, "Covert Channels in the IP Time To Live Field," v zborniku *Australian Telecommunication Networks and Applications Conference (ATNAC)*, Australia, dec. 2006.

Kazalo slik