SADRŽAJ

PREDGOVOR Katie Moussouris **xi**

ZAHVALNICE **xv**

UVOD **xvii**

Zašto da čitate ovu knjigu xviii
Šta se nalazi u knjizi xviii
Kako da koristite knjigu xx
Kontakt xx

1
OSNOVE UMREŽAVANJA **1**

Arhitektura i protokoli mreža 1
Paket internet protokola (IPS) 2
Kapsuliranje podataka 4
Zaglavlja, podnožja i adrese 5
Prenos podataka 6
Rutiranje u mreži 7
Moj model za analizu mrežnih protokola 8
Zaključak 10

2
HVATANJE SAOBRAĆAJA APLIKACIJE **11**

Pasivno hvatanje mrežnog saobraćaja 12
Kratak pregled aplikacije Wireshark 12
Alternativne pasivne tehnike hvatanja saobraćaja 14
Praćenje sistemskih poziva 14
Uslužni program strace u Linuxu 16
Praćenje mrežnih veza pomoću alatke DTrace 16
Process Monitor u Windowsu 18
Prednosti i nedostaci pasivnog hvatanja saobraćaja 19
Aktivno hvatanje mrežnog saobraćaja 20
Mrežni posrednici 21
Posrednik za prosleđivanje priključaka 21
SOCKS posrednik 24
HTTP posrednici 29
Prosleđivanje HTTP posrednika 30
Reverzni HTTP posrednik 32
Zaključak 35

3
STRUKTURE MREŽNIH PROTOKOLA 37

Strukture binarnih protokola . . . 38
Numerički podaci . . . 38
Logičke vrednosti . . . 41
Bit indikatori . . . 42
Binarni format endian . . . 42
Tekstualni i drugi čitljivi podaci . . . 43
Binarni podaci sa promenljivom dužinom . . . 47
Datumi i vremena . . . 50
Vreme u POSIX-u / Unixu . . . 50
FILETIME u Windowsu . . . 50
Obrazac TLV . . . 51
Multipleksiranje i fragmentacija . . . 51
Informacije o mrežnoj adresi . . . 53
Strukturirani binarni formati . . . 53
Strukture tekstualnih protokola . . . 55
Numerički podaci . . . 55
Tekstualne logičke vrednosti . . . 55
Datumi i vremena . . . 56
Podaci sa promenljivom dužinom . . . 56
Strukturirani tekstualni formati . . . 56
Kodiranje binarnih podataka . . . 59
Heksadecimalno kodiranje . . . 59
Sistem Base64 . . . 60
Zaključak . . . 62

4
NAPREDNO HVATANJE SAOBRAĆAJA APLIKACIJE 63

Menjanje putanje saobraćaja . . . 64
Praćenje mrežne putanje . . . 64
Tabele za rutiranje . . . 65
Konfigurisanje rutera . . . 66
Rutiranje u Windowsu . . . 67
Rutiranje u *nix sistemima . . . 67
Prevođenje mrežnih adresa . . . 68
Izvorna NAT tabela . . . 68
Konfigurisanje izvorne NAT tabele u Linuxu . . . 69
Odredišna NAT tabela . . . 70
Prosleđivanje saobraćaja ka mrežnom prolazu . . . 71
Lažiranje DHCP saobraćaja . . . 71
Trovanje ARP protokola . . . 74
Zaključak . . . 78

5
ANALIZA SAOBRAĆAJA 79

Aplikacija koja proizvodi saobraćaj: SuperFunkyChat . . . 80
Pokretanje servera . . . 80
Pokretanje klijenata . . . 81
Komunikacija između klijenata . . . 81

Kratak pregled analize pomoću alatke Wireshark 82
Generisanje mrežnog saobraćaja i hvatanje paketa 83
Osnovna analiza 84
Čitanje sadržaja TCP sesije 85
Identifikovanje strukture paketa pomoću opcije Hex Dump 86
Prikaz pojedinačnih paketa 87
Određivanje strukture protokola 88
Testiranje pretpostavki 89
Seciranje protokola pomoću jezika Python 90
Izrada Wireshark disektora u jeziku Lua 96
Definisanje disektora 98
Seciranje u jeziku Lua 99
Raščlanjivanje paketa poruke 100
Korišćenje posrednika za aktivnu analizu saobraćaja 103
Podešavanje posrednika 103
Analiza protokola pomoću posrednika 106
Dodavanje osnovnog raščlanjivanja protokola 107
Menjanje ponašanja protokola 108
Zaključak 110

6
OBRNUTO INŽENJERSTVO APLIKACIJA 111

Kompajleri, interpreter i asembleri 112
Interpretirani jezici 112
Kompajlirani jezici 113
Statičko i dinamičko povezivanje 113
Arhitektura x86 114
Arhitektura skupa instrukcija 114
Registri procesora 116
Tok programa 119
Osnove operativnih sistema 119
Formati izvršnih datoteka 120
Odeljci 120
Procesi i programske niti 121
Interfejs za umrežavanje operativnog sistema 121
Binarni interfejs za aplikacije 124
Statičko obrnuto inženjerstvo 125
Kratko uputstvo za korišćenje besplatnog disasemblera IDA Pro 126
Analiza promenljivih i argumenata steka 128
Identifikovanje ključnih funkcionalnosti 129
Dinamičko obrnuto inženjerstvo 135
Definisanje tačaka prekida 136
Prozori programa za otkrivanje grešaka 136
Gde se definišu tačke prekida 137
Upravljani jezici obrnutog inženjerstva 138
.NET aplikacije 138
Korišćenje alatke ILSpy 139
Java aplikacije 142
Kako se radi sa maskiranjima 143
Izvori informacija o obrnutom inženjerstvu 144
Zaključak 145

7
BEZBEDNOST MREŽNIH PROTOKOLA 147

Algoritmi za šifrovanje 148
Supstitucione šifre 149
XOR šifrovanje 150
Generatori slučajnih brojeva 151
Šifrovanje simetričnim ključem 151
Blokovske šifre 152
Režimi blokovskog šifrovanja 155
Dopunjavanje u blokovskom šifrovanju 158
Napad „padding oracle” 159
Šifre toka 161
Šifrovanje asimetričnim ključem 162
RSA algoritam 162
RSA dopunjavanje 164
Algoritam Diffie–Hellman za razmenu ključeva 165
Algoritmi za potpisivanje 166
Kriptografski algoritmi za heširanje 167
Asimetrični algoritmi za potpisivanje 167
Kodovi za proveru verodostojnosti poruke 168
Infrastruktura javnog ključa 171
X.509 sertifikati 172
Provera lanca sertifikata 173
Studija slučaja: bezbednost transportnog protokola 174
Usaglašavanje za TLS vezu 175
Početno pregovaranje 176
Provera identiteta u krajnjoj tački 176
Uspostavljanje šifrovanja 178
Ispunjavanje bezbednosnih zahteva 179
Zaključak 180

8
IMPLEMENTIRANJE MREŽNIH PROTOKOLA 181

Ponavljanje postojećeg uhvaćenog mrežnog saobraćaja 181
Hvatanje saobraćaja alatkom Netcat 182
Korišćenje jezika Python za ponovno slanje uhvaćenog UDP saobraćaja 184
Promena namene posrednika za analizu 185
Promena namene postojećeg izvršnog koda 190
Promena namene koda u .NET aplikacijama 191
Promena namene koda u Java aplikacijama 196
Neupravljane izvršne datoteke 198
Šifrovanje i rad sa TLS-om 202
Šifrovanje na delu 202
Dešifrovanje TLS saobraćaja 203
Zaključak 209

9
OSNOVNI UZROCI BEZBEDNOSNE RANJIVOSTI 211

Klase bezbednosne ranjivosti 212
Daljinsko izvršavanje koda 212
Uskraćivanje usluga 212
Obelodanjivanje informacija 213
Zaobilaženje provere identiteta 213
Premošćavanje ovlašćenja 213
Ranjivosti zbog oštećenja u memoriji 214
Bezbedni i nebezbedni programski jezici 214
Prekoračenja memorijskog bafera 215
Indeksiranje bafera izvan granica 220
Napad proširivanjem podataka 221
Otkazi pri dinamičkoj dodeli memorije 222
Podrazumevani akreditivi ili akreditivi upisani u kôd 222
Nabrajanje korisnika 223
Greška tokom pristupa resursima 224
Kanonizovanje 224
Preobimne greške 226
Napadi iscrpljivanjem memorije 227
Napadi iscrpljivanjem skladišta 228
Napadi iscrpljivanjem procesora 229
Algoritamska složenost 229
Podesiva kriptografija 231
Ranjivosti zbog znakovnog niza formata 232
Injektovanje komande 233
SQL injektovanje 233
Zamena znakova za kodiranje teksta 235
Zaključak 236

10
PRONALAŽENJE I ISKORIŠĆAVANJE BEZBEDNOSNIH RANJIVOSTI 237

Faz testiranje 238
Najjednostavniji faz test 238
Faz testiranje sa mutiranjem 239
Generisanje testnih slučajeva 239
Trijaža ranjivosti 240
Aplikacije za otkrivanje grešaka 240
Kako da povećate verovatnoću pronalaženja osnovnog uzroka otkaza 247
Iskorišćavanje najčešćih ranjivosti 250
Iskorišćavanje ranjivosti zbog oštećivanja podataka u memoriji 250
Ranjivost zbog proizvoljnog upisa u memoriju 257
Pisanje koda komandnog okruženja 260
Početak 260
Jednostavna tehnika za otkrivanje grešaka 263
Sistemski pozivi 264
Izvršavanje drugih programa 268
Generisanje koda komandnog okruženja pomoću alatke Metasploit 270
Ublažavanja zloupotreba oštećenja podataka u memoriji 271
Sprečavanje izvršavanja 272
Sprečavanje zloupotrebe povratno orijentisanog programiranja 273

Randomizacija rasporeda adresnog prostora (ASLR) . . . 275
Otkrivanje prekoračenja steka pomoću memorijskih kolačića . . . 278
Zaključak . . . 281

KOMPLETI ALATKI ZA ANALIZU MREŽNIH PROTOKOLA 283

Alatke za pasivno hvatanje saobraćaja i analizu mrežnog protokola . . . 283
Microsoft Message Analyzer . . . 284
TCPDump and LibPCAP . . . 284
Wireshark . . . 285
Aktivno hvatanje i analiza mrežnog saobraćaja . . . 286
Canape . . . 286
Canape Core . . . 287
Mallory . . . 287
Povezivanje i testiranje protokola na mreži . . . 288
Hping . . . 288
Netcat . . . 288
Nmap . . . 288
Testiranje veb-aplikacija . . . 289
Burp Suite . . . 289
Zed Attack Proxy (ZAP) . . . 290
Mitmproxy . . . 290
Radni okviri za faz testiranje, generisanje paketa i iskorišćavanje ranjivosti . . . 291
American Fuzzy Lop (AFL) . . . 291
Kali Linux . . . 292
Metasploit Framework . . . 292
Scapy . . . 293
Sulley . . . 293
Lažiranje i preusmeravanje na mreži . . . 293
DNSMasq . . . 293
Ettercap . . . 293
Obrnuto inženjerstvo izvršnih datoteka . . . 294
Java Decompiler (JD) . . . 294
IDA Pro . . . 295
Hopper . . . 295
ILSpy . . . 296
.NET Reflector . . . 296

SPISAK TERMINA KORIŠĆENIH U KNJIZI 299

INDEKS 303

PREDGOVOR

Kada sam upoznala Jamesa Forshawa, moj posao je u časopisu *Popular Science* iz 2007. godine opisan kao jedan od deset najgorih poslova u oblasti nauke: titula tzv. „Microsoftovih *fizikalaca* za bezbednost" dodeljena je svim članovima centra *Microsoft Security Response Center (MSRC).* Naša radna mesta našla su se na toj listi (proslavljenoj do te mere među nama koji se zlopatimo u Redmondu, država Vašington, da smo napravili i odgovarajuće majice) u istom rangu sa ispitivanjem fekalija kitova, a tek nešto višem od vazektomije slonova. Razlog tome je bila najava čitave gomile izveštaja o bezbednosnim propustima u Microsoftovim proizvodima.

James je moju pažnju prvi put privukao upravo u MSRC-u, kao stručnjak za bezbednosne strategije sa velikim entuzijazmom, kreativnošću i ošrim okom za netipične probleme koje je lako previdеti. James je bio autor najzanimljivijih izveštaja o bezbednosnim propustima – što nije mala stvar, s obzirom na to da u MSRC od istraživača bezbednosti godišnje stiže više od 200.000 izveštaja takve vrste. James je otkrivao ne samo jednostavne programske greške, nego i probleme u platformi .NET

Framework. Greške na nivou osnovne arhitekture teže je rešiti pomoću jednostavnih programskih zakrpa i zato je njihovo otkrivanje više nego dragoceno za Microsoft i njegove klijente.

U junu 2013. godine pokrenula sam prve dodele nagrada za pronađene propuste u kompaniji Microsoft. U prvom paketu nagrada imali smo tri programa, u kojima je istraživačima bezbednosti poput Jamesa trebalo obezbediti novčanu naknadu ako Microsoftu prijave najozbiljnije bezbednosne propuste. Znala sam da ćemo efikasnost tih programa dokazati samo ako se prijave izuzetne bezbednosne greške.

Naravno, pokretanje programa nije nužno značilo da će nalazači grešaka doći kod nas. Borili smo se za najveštije svetske lovce na greške, a bilo je i drugih novčanih nagrada. Osim toga, nisu sva tržišta za za bezbednosne propuste i njihovo iskorišćavanje namenjena odbrani, nego postoje i dobro organizovana nacionalna ili kriminalna napadačka tržišta. Pored toga, Microsoft je računao na one nalazače koji su već dostigli nivo od 200.000 besplatnih godišnjih izveštaja o propustima. Nagrade je trebalo iskoristiti za to da se tim dobronamernim i nesebičnim lovcima na greške skrene pažnja na probleme oko kojih je Microsoftu pomoć bila najpotrebnija.

Zato sam pozvala Jamesa i ostale, verujući da nam mogu isporučiti traženu „robu sa greškom". Prve Microsoftove nagrade žarko smo želeli da dodelimo za ranjivost Internet Explorera 11 beta, naročito za nešto za šta ih nijedan proizvođač softvera nikada nije dodelio: nove metode za iskorišćavanje ranjivosti. Ova poslednja novčana nagrada postala je poznata pod nazivom *Mitigation Bypass Bounty* i iznosila je 100.000 dolara.

Još se sećam kako sam u Londonu uz pivo pokušavala da zagrejem Jamesa za bezbednosne propuste u Internet Exploreru, dok mi je on objašnjavao kako se nije baš previše bavio bezbednošću pregledača i upozorio me da ne očekujem previše.

Na kraju je prijavio četiri jedinstvena izlaza iz karantina za Internet Explorer 11 beta.

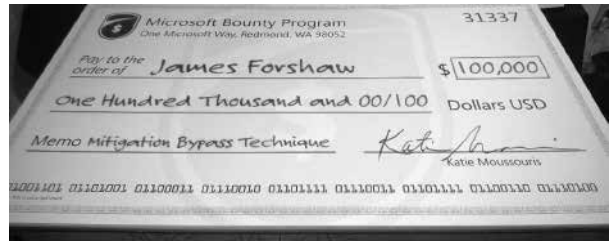
Četiri.

Ti izlazi iz karantina nalazili su se u onim delovima koda Internet Explorera koji su promakli svim našim internim timovima, kao i spoljnim timovima za testiranje neprobojnosti. Izlazi iz karantina su ključni jer pomažu da se na pouzdaniji način iskorišćavaju ostale vrste ranjivosti. James je zaradio nagrade za sve četiri greške, koje mu je isplatio sâm tim Internet Explorera, kao i dodatnih 5000 dolara iz mog budžeta za nagrade – iako mi se sada čini da je zaslužio mnogo više. To zaista nije bio loš rezultat za lovca na greške koji se nikada nije bavio bezbednošću veb-pregledača.

Jednog svežeg jesenjeg dana, samo nekoliko meseci kasnije, pozvala sam Jamesa telefonom i saopštila mu bez daha da je upravo ušao u istoriju. Oduševljena što imam čast da mu prenesem novost, obavestila sam ga da je prihvaćen njegov ulazak u drugi Microsoftov program nagrada za otkrivanje bezbednosnih propusta, *Mitigation Bypass Bounty* za 100.000 dolara. James Forshaw je pronašao nov način da u najnovijem

operativnom sistemu zaobiđe sve odbrane platforme korišćenjem propusta na nivou arhitekture, i tako zasluži Microsoftovu prvu nagradu od 100.000 dolara.

U tom telefonskom razgovoru James mi je rekao da me zamišlja kako mu javno dodeljujem lažni i komično uvećani ček na Microsoftovoj internoj konferenciji *BlueHat*. Odmah posle poziva poslala sam poruku odeljenju za marketing, i tako su James i džinovski ček postali deo istorije Microsofta i interneta.



Ono što će čitaoci sasvim sigurno naći na narednim stranicama ove knjige delići su Jamesove genijalnosti – iste one koju sam uočila pregledajući izveštaje o propustima pre nekoliko godina. Postoji mali broj vrednih istraživača bezbednosti koji mogu da pronađu propuste u jednoj naprednoj tehnologiji, a još je manji broj onih koji mogu jednako dosledno da ih pronađu u više njih. A tu su i ljudi kao što je James Forshaw, koji sa hirurškom preciznošću mogu da se usredsrede na ozbiljnije probleme u arhitekturi. Nadam se da će čitaoci ovu knjigu, kao i svaku buduću koju James bude napisao, posmatrati kao praktičan vodič i izvor inspiracije za svoj budući rad.

Na sastanku povodom dodeljivanja nagrade u Microsoftu, dok su članovi tima Internet Explorera u čudu vrteli glavama, pitajući se kako je moguće da su propustili greške koje je James prijavio, izjavila sam: „James može da vidi i damu u crvenoj haljini i kôd kojim se ona vizuelizuje u filmu *Matriks*." Svi su prihvatili ovo objašnjenje za Jamesov način rada i razmišljanja, jer je reč o čoveku koji snagom uma može da savija kašike. Ako dobro proučite ono što radi i otvorite svoj um, to biste mogli da postignete i vi.

Svima onima koji pronalaze bezbednosne propuste širom sveta: neka ovo postane vaš standard, iako je visok. Svim nebrojenim anonimnim *fizikalcima* za bezbednost softvera: neka svi vaši izveštaji budu dragoceni kao oni koje je napisao jedinstveni i nezamenljivi James Forshaw.

Katie Moussouris,
osnivač i izvršni direktor kompanije Luta Security,
oktobar 2017. godine

1

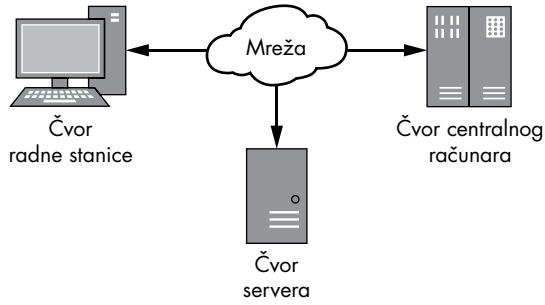
OSNOVE UMREŽAVANJA

Da biste napadali mrežne protokole, treba da razumete osnove umrežavanja računara. Što bolje shvatite kako se tipične mreže grade i funkcionišu, lakše ćete to znanje primeniti na hvatanje mrežnog saobraćaja, analizu i iskorišćavanje ranjivosti novih protokola.

Ovo poglavlje je uvod u osnovne koncepte mreža, sa kojima se svakodnevno srećete pri analizi mrežnih protokola. Ono će vam pomoći da utemeljite način razmišljanja o mrežnim protokolima i da tokom analize lakše pronađete bezbednosne propuste koji su vam ranije bili nepoznati.

Arhitektura i protokoli mreža

Hajde da se prvo pozabavimo osnovnim terminima umrežavanja i postavimo najvažnije pitanje: šta je mreža? *Mreža* (engl. *network*) je skup dvaju ili više računara koji se povezuju da bi delili informacije. Uobičajeno je da se svaki povezani uređaj na mreži naziva čvorom (engl. *node*), kako bi se koncepti mogli primenjivati na razne vrste uređaja. Na slici 1-1. prikazan je jednostavan primer.



Slika 1-1: Jednostavna mreža sa tri čvora

Na slici su prikazana tri čvora povezana u zajedničku mrežu. Svaki čvor može imati drugačiji operativni sistem ili hardver. Međutim, sve dok se na nekom čvoru poštuje skup pravila, ili *mrežni protokol* (engl. *network protocol*), taj čvor može da komunicira sa drugim čvorovima u mreži. Da bi se komunikacija odvijala na ispravan način, svi čvorovi u mreži moraju da poštuju isti mrežni protokol.

Mrežni protokol služi za obavljanje mnogih funkcija, uključujući jednu ili više od sledećih:

Održavanje stanja sesije Protokolima se obično implementiraju mehanizmi za pravljenje novih veza i ukidanje postojećih.

Identifikovanje čvorova preko adresiranja Podaci se moraju prenositi na ispravan čvor mreže. Nekim protokolima se implementira mehanizam za adresiranje kojim se identifikuju određeni čvorovi ili grupe čvorova.

Kontrolisanje toka Količina podataka koja se prenosi kroz mrežu ograničena je. Protokolima se mogu implementirati načini za upravljanje tôkom podataka, kako bi se povećala propusna moć i smanjilo kašnjenje.

Garantovanje redosleda prenesenih podataka Za mnoge mreže se ne može garantovati da će redosled u kome se podaci šalju odgovarati redosledu u kome se oni primaju. Protokol može promeniti redosled podataka i obezbediti njihov ispravan redosled pri isporuci.

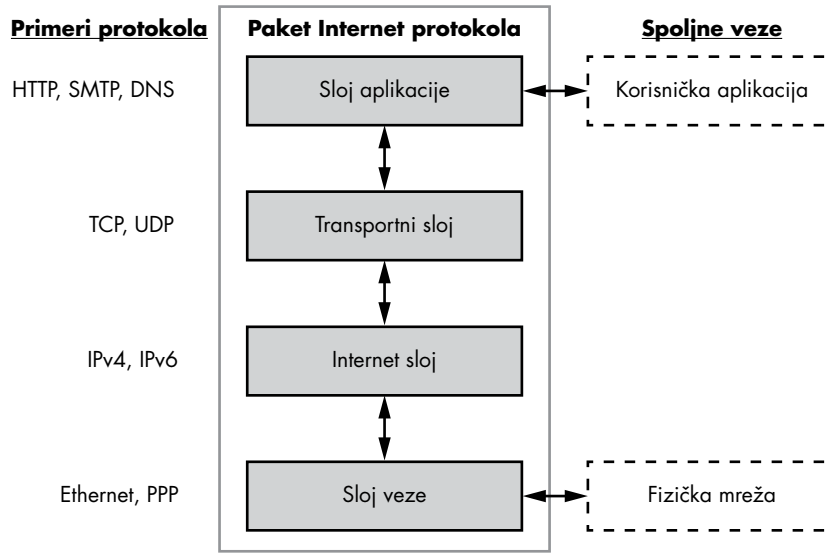
Otkrivanje i ispravljanje grešaka Mnoge mreže nisu potpuno pouzdane pa podaci mogu da se oštete. Važno je da se to oštećenje otkrije i, u idealnom slučaju, ispravi.

Formatiranje i kodiranje podataka Podaci nemaju uvek onaj format koji je pogodan za prenos preko mreže. Protokolom se mogu definisati načini kodiranja podataka, kao što je kodiranje engleskog teksta u obliku binarnih vrednosti.

Paket internet protokola (IPS)

TCP/IP je de facto protokol koji se koristi u savremenim mrežama. Iako TCP/IP možete posmatrati kao jedinstven protokol, on je zapravo kombinacija dvaju protokola: *protokola za kontrolu prenosa* (engl. *Transmission*

Control Protocol, TCP) i *internet protokola* (engl. *Internet Protocol, IP).* Ova dva protokola formiraju deo *paketa internet protokola* (engl. *Internet Protocol Suite, IPS),* konceptualnog modela po kome mrežni protokoli šalju mrežni saobraćaj preko interneta i u kome se mrežna komunikacija može raščlaniti u četiri sloja, kao što je prikazano na slici 1-2.



Slika 1-2: Slojevi paketa internet protokola

Ova četiri sloja formiraju *skup protokola* (engl. *protocol stack*). U nastavku se objašnjava svaki sloj IPS-a:

Sloj veze (sloj 1) Ovo je sloj najnižeg nivoa i njime se opisuju fizički mehanizmi koji se koriste za prenos informacija od jednog do drugog čvora u lokalnoj mreži. U najpoznatijije primere spadaju Ethernet protokol (ožičen i bežični), kao i protokol od tačke do tačke (engl. *Point-to-Point Protocol, PPP*).

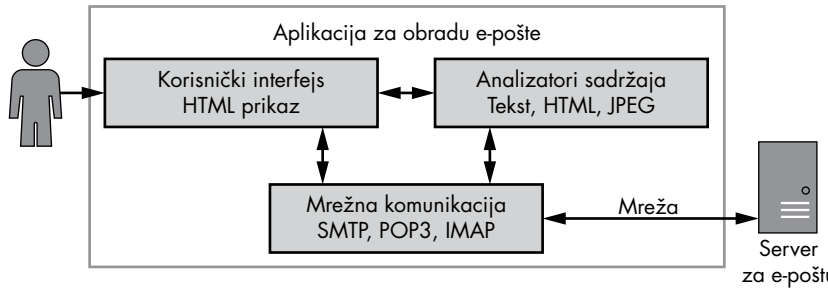
Internet sloj (sloj 2) Ovaj sloj sadrži mehanizme za adresiranje čvorova mreže. Za razliku od sloja 1, čvorovi ne moraju da se nalaze u lokalnoj mreži. Ovaj nivo sadrži IP; stvarni protokoli koji se koriste u savremenim mrežama mogu biti verzija 4 (IPv4) ili verzija 6 (IPv6).

Transportni sloj (sloj 3) Ovaj sloj je odgovoran za veze između klijenata i serverâ, čime se ponekad obezbeđuje ispravan redosled paketa, kao i multipleksiranje usluga. Multipleksiranje usluga omogućava da jedan čvor podrži više različitih usluga tako što se svakoj usluzi dodeljuje drugi broj, i taj broj se naziva *priključak* (engl. *port*). Na ovom sloju rade TCP protokol i *protokol za korisničke datagrame* (engl. *User Datagram Protocol, UDP*).

Sloj aplikacije (sloj 4) Ovaj sloj sadrži mrežne protokole, kao što su *protokol za prenos hiperteksta* (engl. *HyperText Transport Protocol, HTTP),* kojim se prenose sadržaji veb-stranica; *jednostavni protokol za prenos elektronske pošte* (engl. *Simple Mail Transport Protocol, SMTP),*

kojim se prenose elektronska pošta i *sistem imenovanja domena* (engl. *Domain Name System, DNS)*, kojim se ime konvertuje u čvor na mreži. U ovoj knjizi bavićemo se prvenstveno ovim slojem.

Svaki sloj je u interakciji samo sa slojem iznad ili ispod, ali moraju da postoje spoljne interakcije sa skupom. Na slici 1-2. prikazane su dve spoljne veze. Sloj veze je u interakciji sa vezom fizičke mreže, prenoseći na fizički medijum podatke kao što su električni ili svetlosni impulsi. Sloj aplikacije je u interakciji sa aplikacijom korisnika: *aplikacija* (engl. *application*) je skup srodnih funkcionalnosti pomoću kojih se korisniku pružaju usluge. Na slici 1-3. prikazan je primer aplikacije za obradu elektronske pošte. Usluga koja se pruža ovakvom aplikacijom jeste slanje i primanje poruka preko mreže.



Slika 1-3: Primer aplikacije za obradu elektronske pošte

Aplikacija se obično sastoji od sledećih komponenata:

Mrežna komunikacija Pomoću ove komponente odvija se komunikacija preko mreže i obrađuju dolazni i odlazni podaci. Za aplikacije za obradu e-pošte najčešće se koriste standardni protokoli kao što su SMTP ili POP3.

Analizatori sadržaja Sadržaj podataka koji se prenose preko mreže najčešće mora da se izdvoji i obradi. Takav sadržaji može da obuhvati tekstualne podatke, kao što je telo e-poruke, ili slike odnosno video zapise.

Korisnički interfejs (engl. *user interface, UI*) UI omogućava da korisnik pregleda primljenu e-poštu i sastavlja nove e-poruke za prenos. U aplikaciji za obradu e-pošte, UI omogućava prikaz e-poruka u veb-čitaču pomoću HTML-a.

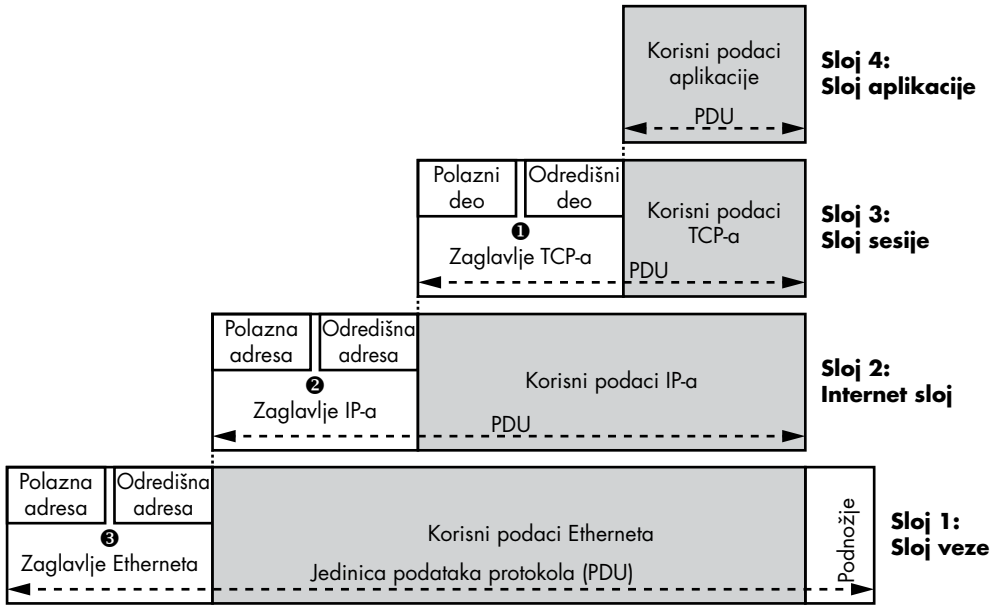
Imajte na umu da korisnik koji komunicira sa korisničkim interfejsom ne mora biti čovek. To može biti i neka druga aplikacija, kojom se automatizuje slanje i primanje e-pošte preko alatke sa komandnom linijom.

Kapsuliranje podataka

Svaki sloj u paketu internet protokola gradi se na sloju ispod i može da kapsulira podatke iz sloja iznad tako da oni mogu da se premeštaju iz jednog sloja u drugi. Podaci koje prenosi svaki sloj nazivaju se *jedinica podataka protokola* (engl. *protocol data unit, PDU).*

Zaglavlja, podnožja i adrese

PDU u svakom sloju sadrži korisne podatke koji se prenose. Uobičajeno je da se na početak korisnih podataka stavi *zaglavlje* (engl. *header*), koje sadrži informacije potrebne za prenos takvih podataka, kao što je *adresa* (eng. *address*) polaznih i odredišnih čvorova na mreži. Ponekad jedinica podataka protokola sadrži i *podnožje* (engl. *footer*), koje se dodaje na kraj korisnih podataka i sadrži vrednosti pomoću kojih se obezbeđuje ispravan prenos, kao što su informacije o otkrivanju grešaka. Na slici 1-4. prikazano je kako su jedinice podataka protokola raspoređene u paketu internet protokola.



Slika 1-4: Kapsuliranje podataka u IPS-u

Zaglavlje TCP-a sadrži broj polaznog i odredišnog priključka ❶. Ti brojevi priključaka omogućavaju da jedan čvor ima više jedinstvenih mrežnih veza. Brojevi priključaka za TCP (i UDP) kreću se u rasponu od 0 do 65535. Većina brojeva priključaka se po potrebi dodeljuje novim vezama, ali neki brojevi imaju posebna zaduženja, kao što je priključak 80 za HTTP. (Aktuelnu listu dodeljenih brojeva priključaka možete pronaći u datoteci */etc/services* na većini operativnih sistema srodnih Unixu.) Korisni podaci TCP-a i zaglavlje zajedno se nazivaju *segment*, dok se korisni podaci UDP-a i zaglavlje nazivaju *datagram*.

IP koristi polaznu i odredišnu adresu ❷. *Odredišna adresa* (engl. *destination address*) omogućava da se podaci šalju na poseban čvor u mreži. *Polazna adresa* (engl. *source address*) omogućava da prijemnik podataka zna koji je čvor poslao podatak i da prijemnik odgovori pošiljaocu.

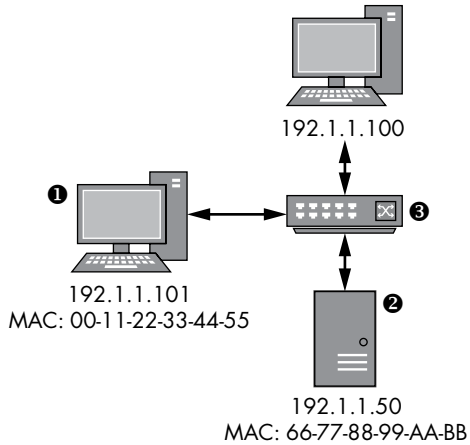
IPv4 koristi 32-bitne adrese, što ćete najčešće viđati u obliku zapisa od četiri broja razdvojena tačkama, kao što je 192.168.10.1. IPv6 koristi 128-bitne adrese, jer 32-bitne adrese nisu dovoljne za broj čvorova

na savremenim mrežama. IPv6 adrese se obično pišu u heksadecimlnim brojevima razdvojenih dvotačkama, kao što je fe80:0000:0000:0000:897b:581e:44b0:2057. Dugi znakovni nizovi sa brojevima 0000 sažimaju se u dve dvotačke. Na primer, prethodna IPv6 adresa se može napisati i kao fe80::897b:581e:44b0:2057. Korisni podaci IP-a i zaglavlje zajednički se nazivaju *paket* (engl. *packet*).

I Ethernet sadrži polazne i odredišne adrese ❸. Ethernet koristi 64-bitnu vrednost koja se naziva MAC adresa, ili *adresa za upravljanje pristupom medijumima* (engl. *Media Access Control*), a koja se obično zadaje tokom proizvodnje Ethernet adaptera. MAC adrese se najčešće pišu kao nizovi heksadecimalnih brojeva razdvojenih crtama ili dvotačkama, kao što je 0A-00-27-00-00-0E. Korisni podaci Ethernet-a, uključujući zaglavlje i podnožje, zajednički se nazivaju *okvir* (engl. *frame*).

Prenos podataka

Opišimo ukratko kako se podaci prenose sa jednog čvora na drugi pomoću modela za kapsuliranje podataka IPS-a. Na slici 1-5. prikazuje se jednostavna Ethernet mreža sa tri čvora.



Slika 1-5: Jednostavna Ethernet mreža

U ovom primeru, čvor ❶ sa IP adresom 192.1.1.101 želi da pošalje podatke pomoću IP-a na čvor ❷ sa IP adresom 192.1.1.50. (*Komutator* (engl. *switch*) ❸ prosleđuje Ethernet okvire među svim čvorovima u mreži. Komutatoru nije potrebna IP adresa, jer radi samo u sloju veze.) Evo šta je potrebno da bi se podaci poslali sa jednog čvora na drugi:

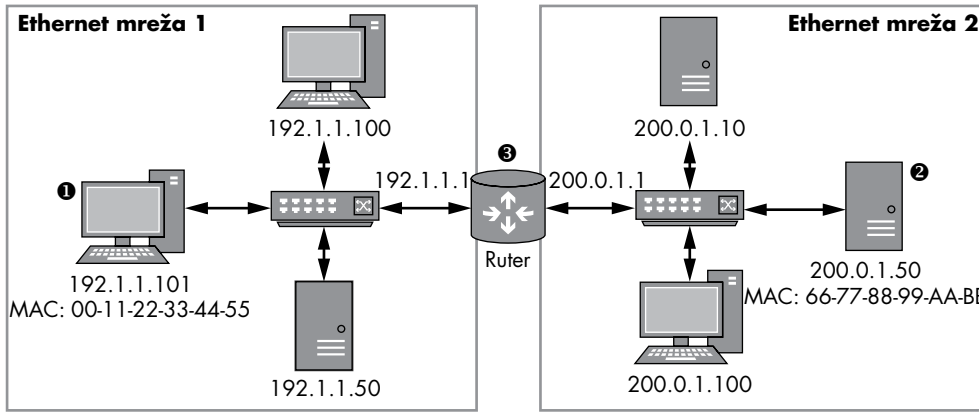
1. Čvor ❶ mrežnog skupa operativnog sistema kapsulira podatke aplikacije i transportnog sloja, i pravi IP paket sa polaznom adresom 192.1.1.101 i odredišnom adresom 192.1.1.50.
2. U ovoj tački operativni sistem može da kapsulira IP podatke kao Ethernet okvir, ali ne mora da zna MAC adresu odredišnog čvora. On može da zatraži MAC adresu za određenu IP adresu pomoću

protokola za razrešavanje adresa (engl. *Address Resolution Protocol, ARP*), koji šalje zahtev svim čvorovima mreže da nađu MAC adresu za odredišnu IP adresu.

3. Kada čvor ❶ primi ARP odgovor, on može da napravi okvir, zadajući lokalnu MAC adresu 00-11-22-33-44-55 kao polaznu, a 66-77-88-99-AA-BB kao odredišnu adresu. Novi okvir se prenosi preko mreže i prima ga komutator ❸.
4. Komutator prosleđuje okvir odredišnom čvoru, koji će raspakovati IP paket i potvrditi da je odredišna IP adresa odgovarajuća. Nakon toga se korisni podaci IP-a izdvajaju i prosleđuju u skup, kako bi ih primila aplikacija koja ih očekuje.

Rutiranje u mreži

Za Ethernet je potrebno da svi čvorovi budu direktno povezani u istu lokalnu mrežu. Taj zahtev predstavlja glavno ograničenje za stvarnu globalnu mrežu, jer nije praktično da se svaki čvor fizički povezuje sa svim ostalim čvorovima. Umesto da se traži da svi čvorovi budu direktno povezani, polazne i odredišne adrese omogućavaju da se podaci *rutiraju* (engl *routing*) kroz različite mreže, sve dok ne stignu na traženi odredišni čvor, kao što je prikazano na slici 1-6.



Slika 1-6: Primer rutirane mreže u kojoj su povezane dve Ethernet mreže

Na slici 1-6. prikazane su dve Ethernet mreže, svaka sa zasebnim opsegom IP mrežnih adresa. U nastavku se opisuje kako IP koristi ovaj model da pošalje podatke iz čvora ❶ mreže 1 u čvor ❷ mreže 2.

1. Čvor ❶ mrežnog skupa operativnog sistema kapsulira podatke aplikaciju i transportnog sloja, i pravi IP paket sa polaznom adresom 192.1.1.101 i odredišnom adresom 200.0.1.50.
2. Mrežni skup treba da pošalje Ethernet okvir, ali zbog toga što odredišna IP adresa ne postoji ni na jednoj Ethernet mreži sa kojom je čvor povezan, mrežni skup proverava *tabelu za rutiranje* (engl. *routing*

table) svog operativnog sistema. U ovom primeru, tabela za rutiranje sadrži element za IP adresu 200.0.1.50. Element pokazuje da ruter ❸ na IP adresi 192.1.1.1 zna kako da stigne na tu odredišnu adresu.

3. Operativni sistem koristi ARP da potraži MAC adresu rutera na adresi 192.1.1.1 i prvobitni IP paket se kapsulira u Ethernet okvir sa tom MAC adresom.
4. Ruter će primiti Ethernet okvir i raspakovati IP paket. Kada ruter proveri odredišnu IP adresu, utvrdiće da IP paket nije upućen ruteru, nego drugom čvoru u drugoj povezanoj mreži. Ruter će potražiti MAC adresu na adresi 200.0.1.50, kapsulirati prvobitni IP paket u novi Ethernet okvir i poslati ga u mrežu 2.
5. Odredišni čvor će primiti Ethernet okvir, raspakovati IP paket i obraditi njegov sadržaj.

Ovaj proces rutiranja može se ponoviti više puta. Na primer, da ruter nije bio direktno povezan s mrežom koja sadrži čvor 200.0.1.50, on bi pogledao u svoju tabelu za rutiranje i odredio sledeći ruter koji može da pošalje IP paket.

Jasno je da bi bilo nepraktično kada bi svaki čvor u mreži znao kako da stigne do svakog drugog čvora na internetu. Ako nema eksplicitno definisanog odredišnog elementa za rutiranje, operativni sistem šalje podrazumevani element tabele za rutiranje, koji se naziva *podrazumevani mrežni prolaz* (engl. *default gateway*) i sadrži IP adresu rutera koji može da prosledi IP pakete na odredište.

Moj model za analizu mrežnih protokola

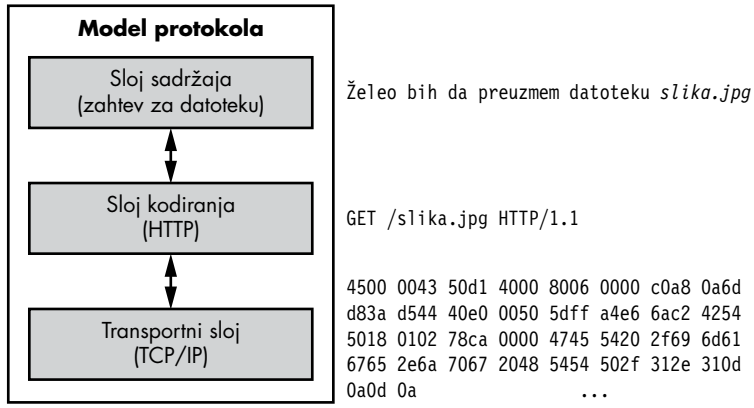
Paketom internet protokola opisuje se kako radi mrežna komunikacija; međutim, za potrebe analize, većina IPS modela nije relevantna. Ponašanje mrežnog protokola aplikacije lakše ćete razumeti ako upotrebite moj model. On sadrži tri sloja (slika 1-7), kojima se ilustruje način na koji bih ja analizirao neki HTTP zahtev.

U nastavku su tri sloja mog modela:

Sloj sadržaja Sadrži značenje predmeta komunikacije. Na slici 1-7. to podrazumeva formiranje HTTP zahteva za datoteku *slika.jpg*.

Sloj kodiranja Sadrži pravila za upravljanje načinom predstavljanja sadržaja. U ovom primeru, HTTP zahtev se kodira kao HTTP GET zahtev, u kome se zadaje datoteka koja se preuzima.

Transportni sloj Sadrži pravila za upravljanje načinom prenosa podataka sa jednog čvora na drugi. U ovom primeru, HTTP GET zahtev se šalje preko TCP/IP veze na priključak 80 udaljenog čvora.

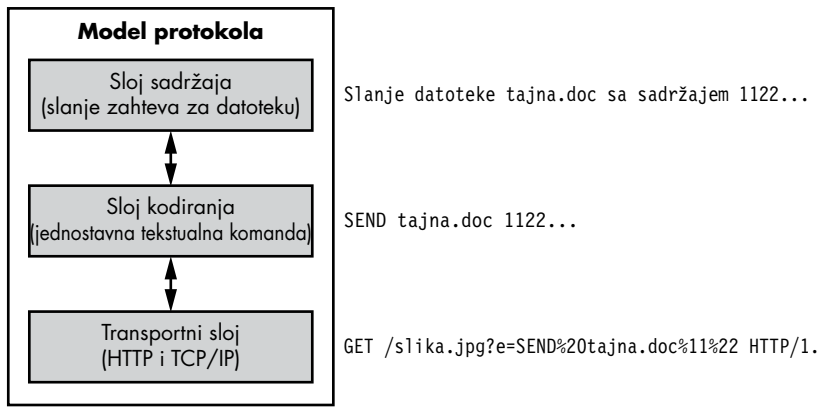


Slika 1-7: Moj konceptualni model protokola

Kada se model ovako raščlani, smanjuje se složenost protokola karakterističnih za aplikacije i omogućava da se iz mrežnog protokola izdvoje detalji koji nisu relevantni. Na primer, pošto nas ovde ne zanima kako se TCP/IP šalje na udaljeni čvor (podrazumevamo da će tamo već nekako stići), posmatramo TCP/IP podatke kao binarni prenos koji funkcioniše.

Da biste shvatili zašto je koristan ovaj model protokola, pogledajte sledeći primer: zamislite da proveravate da li je mrežni saobraćaj na udaru zlonamernog softvera. Otkrili ste da zlonamerni softver koristi HTTP da dobije komande od operatera preko servera. Na primer, operater može zatražiti da zlonamerni softver nabroji sve datoteke na zaraženom disku. Spisak datoteka se može poslati nazad na server i u toj fazi operater može zatražiti da se otpremi određena datoteka.

Ako bismo protokol analizirali iz perspektive načina na koji bi operator komunicirao sa zlonamernim softverom, kao što je zahtev za otpremanje datoteke, novi protokol bi se raščlanio u slojeve kao na slici 1-8.



Slika 1-8: Konceptualni model za protokol zlonamernog softvera korišćenjem HTTP-a

U nastavku se objašnjava svaki sloj novog modela protokola:

Sloj sadržaja Zlonamerna aplikacija šalje serveru ukradenu datoteku pod nazivom *tajna.doc*.

Sloj kodiranja Kodiranje komande za slanje ukradene datoteke jednostavan je tekstualni niz sa komandom `SEND`, iza koje slede ime datoteke i njeni podaci.

Transportni sloj Protokol koristi parametar HTTP zahteva za prenos komande. On se služi standardnim mehanizmom kodiranja znakovima za procenat, čime mu daje karakteristiku legalnog HTTP zahteva.

Obratite pažnju na to da u ovom primeru nismo uzimali u obzir slanje HTTP zahteva preko TCP/IP protokola; kombinovali smo kodiranje i transportni sloj sa slike 1-7. u transportni sloj na slici 1-8. Iako zlonamerni softver i dalje koristi protokole nižeg nivoa, kao što je TCP/IP, ovi protokoli nisu važni za analizu komande zlonamernog softvera za slanje datoteke. Razlog je taj što HTTP preko TCP/IP-a možemo da posmatramo kao jedan transportni sloj koji funkcioniše i usredsredimo se samo na jedinstvene komande zlonamernog softvera.

Ako posmatranje svedemo samo na one slojeve protokola koje treba da analiziramo, izbeći ćemo suvišan posao i usmeriti se na jedinstvene aspekte protokola. S druge strane, da smo ovaj protokol analizirali korišćenjem slojeva sa slike 1-7, mogli bismo pretpostaviti da zlonamerni softver zahteva samo datoteku *slika.jpg*, jer bi izgledalo da je to sve što se HTTP zahtevom obavlja.

Zaključak

U ovom poglavlju izložen je kratak pregled osnova umrežavanja. Bilo je reči o paketu internet protokola (IPS), uključujući protokole na koje ćete nailaziti u stvarnim mrežama, i opisano je kako se podaci prenose s jednog čvora na drugi u lokalnoj mreži i udaljenim mrežama preko rutiranja. Opisan je i način na koji možete da posmatrate mrežne protokole aplikacija, kako biste se lakše usmerili na jedinstvene funkcije protokola i ubrzali analizu.

U poglavlju 2, osnove umrežavanja ćemo koristiti da bismo savladali hvatanje mrežnog saobraćaja radi analize. Cilj hvatanja mrežnog saobraćaja jeste pristup podacima koji su vam potrebni da biste započeli analizu, utvrdili koji se protokoli koriste, i na kraju otkrili bezbednosne propuste koje možete iskorišćavati da biste pomoću tih protokola provaljivali u aplikacije.