UNIVERZITET U BEOGRADU ELEKTROTEHNIČKI FAKULTET



APLIKACIJA ZA PRIKAZ REZULTATA ANALIZE MREŽNOG SAOBRAĆAJA

Master rad

Mentor: doc. dr Zoran Čiča Kandidat: Marija Milojković 2013/3040

Beograd, Septembar 2016.

Sadržaj

SA	ADRŽAJ	2
1.	UVOD	4
2.	. ANALIZIRANJE PAKETA I MREŽNE OSNOVE	5
	 2.1. ANALIZA PAKETA	
3	 2.4.3. Enkapsulacija podataka 2.4.4. Mrežni uređaji 2.4.5. Klasifikacija saobraćaja 	
5.	3.1. <i>PROMISCUOUS</i> MOD 3.2. PRIKUPLJANJE PAKETA U HUB OKRUŽENJU 3.3. PRIKUPLJANJE PAKETA U SVIČ OKRUŽENJU 3.3.1. <i>Port Mirroring</i> 3.3.2. Hubbing Out 3.3.3. Using a tap 3.3.4. <i>ARP cache poisoning</i> 3.4. POZICIONIRANJE ANALIZATORA U SVIČ OKRUŽENJU 3.5. PRIKUPLJANJE PAKETA U RUTER OKRUŽENJU	13 13 13 14 14 15 16 16 16 18 19 20
4.	PROTOKOLI NIŽIH SLOJEVA OSI MODELA	
	 4.1. ARP (ADDRESS RESOLUTION PROTOCOL)	22 24 25 25 26 26 26 26 27 27 27 27 27 28 29 29 29 30 30 30
5.	PROTOKOLI VIŠIH SLOJEVA OSI MODELA	
	 5.1. DHCP (DYNAMIC HOST CONFIGURATION PROTOCOL) 5.1.1. Struktura DHCP paketa 5.1.2. DHCP proces obnavljanja 5.1.3. DHCP tipovi poruka 5.2. DNS (DOMAIN NAME SYSTEM) 5.2.1. Struktura DNS paketa 	32 32 33 33 34 34 34 34
	5.5. III IF (IIIFERIEAI I KANSFER F KUTUCUL)	

6. PF	ROGRAMSKI ALAT WIRESHARK	
6.1.	OSNOVNE OPCIJE WIRESHARK PROGRAMSKOG ALATA	
6.2.	RAD SA SNIMLJENIM PAKETIMA U WIRESHARK PROGRAMSKOM ALATU	
6.3.	NAPREDNE OPCIJE WIRESHARK PROGRAMSKOG ALATA	
7. RI	EZULTATI ANALIZE SNIMLJENOG SAOBRAĆAJA	53
7.1.	ANALIZA PAKETA ARP PROTOKOLA	
7.2.	ANALIZA PAKETA IP PROTOKOLA	
7.3.	ANALIZA PAKETA TCP PROTOKOLA	
7.4.	ANALIZA PAKETA UDP PROTOKOLA	
7.5.	ANALIZA PAKETA ICMP PROTOKOLA	
7.6.	ANALIZA PAKETA DHCP PROTOKOLA	
7.7.	ANALIZA PAKETA DNS PROTOKOLA	74
7.8.	ANALIZA PAKETA HTTP PROTOKOLA	
8. ZA	AKLJUČAK	
LITERA	ATURA	
A. PF	RILOZI	
A.1.	LUA SKRIPTA ZA REGISTROVANJE PORTOVA 4889-4893 ZA HTTP PROTOKOL	
A.2.	LUA SKRIPTA ZA BROJANJE PAKETA KOJI DOLAZE OD/DO IP ADRESE 10.0.33.7	
A.3.	LUA SKRIPTA ZA DEFINISANJE PROTOKOLA DNS NA PORTU 65333	
A 4	LUA SKRIPTA ZA STATISTIČKU ANALIZU PROTOKOLA	90

1.Uvod

Analiza mrežnog saobraćaja predstavlja bitnu stavku prilikom održavanja bilo koje mreže.Jedini način da mrežni administratori obezbede raspoloživost, brzinu i efikasnost u mreži ogleda se u neprekidnom praćenju ponašanja mreže i mrežnog saobraćaja.Ukoliko primete nepravilnosti u radu mreže, sporiji saobraćaj, bilo kakvo sumnjivo ponašanje, kroz analizu snimljenog saobraćaja može da se otkrije uzrok problema.

Svi problemi koji mogu nastati u jednoj mreži polaze od nivoa paketa, gde i aplikacije koje najbolje izgledaju na oko mogu otkriti loše implementacije i gde se naizgled pouzdani protokoli mogu pokazati kao zlonamerni. Kako bismo bolje razumeli probleme do kojih može doći u jednoj mreži, ići ćemo na nivo paketa. Na nivou paketa nema pravih tajni, osim onih enkriptovanih. Što više možemo da uradimo na nivou paketa, to više možemo da kontrolišemo mrežu i rešimo probleme.

U ovom radu korišćen je programski alat Wireshark koji pruža dosta mogućnosti kada je reč o snimanju paketa i analiziranju snimljenih paketa.U poglavlju 2. opisane su osnove analiziranja paketa i navedene mrežne osnove neophodne za razumevanje saobraćaja, paketa i rezultata samog snimanja saobraćaja.U poglavlju 3. je objašnjen sam proces snimanja i prikupljanja paketa u različitim okruženjima.U poglavlju 4. dat je pregled protokola nižih slojeva OSI modela, dok je u poglavlju 5. dat pregled protokola viših slojeva OSI modela, uz najbitnije karakteristike i izgled zaglavlja svakog od njih.Kratak opis Wireshark programskog alata uz osnovne i napredne opcije koje mogu da se koriste prilikom snimanja i analiziranja saobraćaja dati su u poglavlju 6.U poglavlju 7. dat je prikaz rezultata snimanja saobraćaja, uz analizu samih rezultata i grafički prikaz. Poglavlje 8. rezimira rad, dok su u prilozima dati kompletni kodovi skripti koje predstavljaju glavni doprinos ovog rada.

2. Analiziranje paketa i mrežne osnove

Postoji hiljade različitih problema koji se svakodnevno mogu desiti u računarskoj mreži – od jednostavne *spyware* infekcije do kompleksne konfiguracione greške na ruteru, a ponekad je nemoguće rešiti sve te probleme istog trenutka kada se oni dese. Najbolje što možemo da uradimo jeste da se u potpunosti pripremimo za rešavanje problema, u smislu pribavljanja odgovarajućeg znanja i alata.

Svi problemi koji mogu nastati u jednoj mreži polaze od nivoa paketa, gde i aplikacije koje najbolje izgledaju na oko mogu otkriti loše implementacije i gde se naizgled pouzdani protokoli mogu pokazati kao zlonamerni. Kako bismo bolje razumeli probleme do kojih može doći u jednoj mreži, ići ćemo na nivo paketa. Na nivou paketa nema pravih tajni, osim onih enkriptovanih. Što više možemo da uradimo na nivou paketa, to više možemo da kontrolišemo mrežu i rešimo probleme.

2.1. Analiza paketa

Analiza paketa, često nazivana i *packet sniffing* ili analiza protokola opisuje proces snimanja i interpretacije živih podataka koji se razmenjuju kroz mrežu kako bismo što bolje razumeli šta se zaista događa u mreži. Analiza paketa se radi uz pomoć alata za snimanje i analizu, koji prikupljaju pakete podataka koji prolaze kroz mrežu.

Analiziranje paketa pomaže u sledećem:

- Razumevanju mrežnih karakteristika
- Proveri ko je sve na mreži
- Utvrđivanje ko ili šta koristi dostupni protok
- Identifikovanje pikova kada se mreža najviše koristi
- Identifikovanje mogućih napada ili zlonamernih aktivnosti
- Pronalaženje nesigurnih aplikacija

Postoji više tipova programa koji se koriste za analizu paketa, bilo besplatnih, bilo komercijalnih. Par popularnih programa za analizu paketa su *tcpdump*, *OmniPeek* i *Wireshark*. *Tcpdump* je program komandne linije. *OmniPeek* i *Wireshark* imaju GUI (*Graphical User Interfaces*).

2.2. Procena programa za snimanje i analizu paketa

Prilikom odabira programa za analizu paketa potrebno je uzeti u obzir sledeće faktore:

 Podržani protokoli – svi programi za analizu paketa mogu da interpretiraju više različitih protokola. Većina njih mogu da interpretiraju najčešće mrežne protokole (kao što su IPv4 i ICMP), transportne protokole (kao što su TCP i UDP), kao i aplikacione protokole (kao što su DNS i HTTP). Međutim, postoji mogućnost da ne podržavaju netradicionalne ili novije protokole kao što su IPv6, SMBv2 i SIP. Prilikom odabira programa za analizu paketa potrebno je proveriti da li podržavaju sve protokole od interesa.

- User-friendliness Potrebno je uzeti u obzir izgled samog programa, da li je lak za upotrebu i instalaciju i generalni tok standardnih operacija. Program bi trebalo da odgovara nivou stučnosti lica koje će ga koristiti. Za one sa manje iskustva sa analiziranjem paketa, poželjnije je da se ne koriste napredniji*command-line* alat kao što je tepdump. Sa druge strane, iskusni korisnici će verovatno želeti da koriste napredniji program. Ponekad je korisno kombinovanje više programa za analizu paketa za pouzdanije i preciznije podatke, ali za to je ipak neophodno iskustvo.
- Cena Velika prednost mnogih besplatnih programa za analizu paketa je u tome što mogu da stanu rame uz rame sa komercijalnim proizvodima. Najprimetnija razlika između komercijalnih proizvoda i njihovih besplatnih alternativa je u načinu na koji prikazuju rezultate analize. Komercijalni programi tipično nude modernije module koji nedostaju ili nisu konzistentni u besplatnim aplikacijama.
- Programska podrška Bez obzira na stručnost i ekspertizu osobe koja radi analiziranje paketa, povremeno se javlja potreba za podrškom u rešavanju novih problema. Dostupna podrška se ogleda u dokumentaciji developera, javnih foruma i mejling lista. Iako možda ne postoji podrška samih developera za Wireshark, ljudi koji koriste ovu aplikaciju najčešće mogu da nadomeste taj nedostatak. Korisnici programa Wireshark pružaju podršku kroz diskusione panele, wiki dokumentaciju i blogove.
- Podrška za operativne sisteme Nažalost, svi programi za analizu paketa ne podržavaju sve operativne sisteme. Potrebno je da izaberete onaj program koji će raditi na svim operativnim sistemima koje koristite. Pored toga, potrebno je da imate na umu da ćete ponekad snimati (prikupljati) pakete na jednoj mašini, a pregledati te pakete na drugoj. Ponekad će se desiti da ćete morati da koristite različite aplikacije na svakom uređaju, upravo zbog razlika u operativnim sistemima.

2.3. Kako rade programi za snimanje i analizu paketa

Svaki program za snimanje i analizu paketa integriše u sebi kooperativnost između softvera i hardvera. Sam proces rada programa se može podeliti na tri koraka:

- Prikupljanje paketa u prvom koraku, program prikuplja binarne podatke sa mreže. To se tipično radi tako što se odabrani mrežni interfejs prebaci u *promiscuous mode*. Pomenuti način rada omogućava mrežnoj kartici da prisluškuje sav saobraćaj u tom delu mreže, a ne samo saobraćaj koji je direktno adresiran na nju.
- 2) Konverzija u ovom koraku prikupljeni binarni podaci se konvertuju u čitljiviju formu. Na ovom koraku se najčešće završava rad najnaprednijih programa komandne linije. Mrežni podaci mogu da se interpetiraju na najosnovnijem nivou, prepuštajući veći deo analize krajnjem korisniku.
- 3) Analiza treći i poslednji korak uključuje i samu analizu prikupljenih i konvertovanih podataka. Na osnovu informacija koje preuzima iz samog paketa, program verifikuje protokol koji se koristi i počinje analizu specifičnih karakteristika određenih protokola.

2.4. Princip komunikacije između računara

Kako bismo u potpunosti razumeli analizu paketa, prvo moramo razumeti kako računari komuniciraju među sobom. U ovom odeljku ćemo obraditi osnove mrežnih protokola, OSI (Open

Systems Interconnection) model, okvire (*frames*) mrežnih podataka i hardverske komponente koji sve to podržavaju.

2.4.1. Protokoli

Moderne mreže se sastoje od više različitih sistema koji rade na različitim platformama. Kako bi komunikacija bila olakšana, koriste se protokoli. Protokoli predstavljaju skup pravila po kojim se vrši razmena informacija između entiteta u komunikacionim sistemima. Najčešći protokoli su: TCP (*Transmission Control Protocol*), IP (*Internet Protocol*), ARP (*Address Resolution Protocol*), DHCP (*Dynamic Host Configuration Protocol*)...

Protokol može da bude ekstremno jednostavan ili veoma kompleksan, u zavisnosti od funkcija koje nudi. Iako se većina protokola drastično razlikuje, mnogi protokoli daju odgovore na sledeća pitanja:

- Inicijacija uspostave veze da li klijent ili server iniciraju uspostavu veze, kojeinformacije moraju da se razmene pre same komunikacije...
- **Pregovaranje o karakteristikama veze** da li je komunikacija šifrovana, kako se razmenjuju ključevi za šifrovanje između hostova koji komuniciraju...
- Formatiranje podataka na koji način su podaci sadržani u paketu, kojim redosledom se obrađuju podaci od strane uređaja koji ih prima...
- **Detekcija i otklanjanje grešaka** šta se događa ako je paketu potrebno previše vremena da stigne do svoje destinacije, kako se klijent oporavlja ako ne može da uspostavi komunikaciju sa serverom u kratkom vremenskom periodu...
- **Prekid veze** –kako jedan host obaveštava drugog hosta sa kojim se uspostavio vezu da je došlo do kraja komunikacije, koje informacije moraju da se razmene kako bi se prekinula komunikacija tako da obe strane to znaju...

2.4.2. OSI referentni model

OSI referentni model je definisala ISO (*International Organization for Standardization*) 1983. godine. OSI model deli proces mrežne komunikacije na sedam različitih slojeva, kao što je prikazano na slici2.4.1.

Aplikacioni sloj na vrhu OSI modela predstavlja same programe koji se koriste za pristup mrežnim resursima. Fizički sloj na dnu OSI modela je sloj kroz koji putuje stvarni saobraćaj. Protokoli svakog sloja rade zajedno kako bi osigurali da se pravilno rukuje podacima od strane protokola iznad i ispod.



Slika 2.4.1. OSI referentni model [1]

Svaki OSI sloj ima specifične funkcije:

- Aplikacioni sloj (Application Layer / Layer 7) Najviši sloj OSI modela pruža mogućnost korisnicama da zaista pristupe mrežnim resursima. To je jedini sloj koji krajnji korisnici mogu da vide tako što pruža interfejs kao bazu svih mrežnih aktivnosti.
- Sloj prezentacije (*Presentation Layer / Layer 6*) Ovaj sloj transformiše podatke koje dobije u format koji može da pročita aplikacioni sloj. Kodiranje i dekodiranje podataka zavisi od protokola aplikacionog sloja koji prima ili šalje podatke. Sloj prezentacije je zadužen za enkripciju i dekripciju koje se koriste kako bi se osigurali podaci koji se prenose.
- Sloj sesije (Session Layer/ Layer 5) Ovaj sloj je zadužen za dijalog, odnosno sesiju između dva računara. On uspostavlja, rukovodi i raskida vezu između uređaja koji međusobno komuniciraju. Sloj sesije je takođe odgovoran za utvrđivanje da li je veza pun dupleks ili poludupleks, kao i za komunikaciju između hostova u situacijama kada se veza raskida uz potvrdu obe strane koje komuniciraju.
- Transportni sloj (Transport Layer/Layer 4) Glavna uloga transportnog sloja je pružanje pouzdanog prenosa podataka za slojeve iznad njega u OSI modelu. Kroz kontrolu protoka, segmentaciju/desegmentaciju i kontrolu grešaka, transportni sloj osigurava da podaci stignu od jednog do drugoghosta bez grešaka. Transportni sloj koristi i protokole koji uspostavljaju vezu, kao i protokole koji ne uspostavljaju vezu pre prenosa podataka.
- Mrežni sloj (Network Layer/ Layer 3) Ovaj sloj je odgovoran za rutiranje podataka između fizičkih mreža i jedan je od najkompleksnijih u OSI modelu. Zadužen je za logičko adresiranje mrežnih hostova (npr. kroz IP adresiranje), kao i za fragmentaciju paketa i u nekim slučajevima za detekciju greške. Ruteri (routers) su uređaji mrežnog sloja.
- Sloj linka za podatke (*Data Link Layer/ Layer 2*) Ovaj sloj omogućava sredstva za transportovanje podataka preko fizičke mreže. Osnovna uloga sloja linka za podatke je pružanje šeme adresiranja koja se koristi za identifikaciju fizičkih uređaja (npr. MAC adrese). Mostovi (*bridges*) i svičevi (*switches*)su uređaji drugog sloja.
- Fizički sloj (*Physical Layer/ Layer 1*) sloj na dnu OSI modela predstavlja fizički medijum kroz koji se prenose mrežni podaci. Fizički sloj definiše fizičku i električnu prirodu celokupnog hardvera koji se koristi, uključujući hubove, mrežne adaptere,

ripitere, kao i specifikaciju kablova. Ovaj sloj uspostavlja i raskida veze, pruža sredstva za deljenje komunikacionih resursa i vrši konverziju signala iz digitalnog u analogni i obrnuto.

U tabeli 2.4.1. dat je pregled najčešćih protokola koji se koriste za svaki pojedinačni sloj OSI modela.

SLOJ	PROTOKOLI
Aplikacioni sloj	HTTP, SMTP, FTP, Telnet, DNS
Sloj prezentacije	ASCI, MPEG, JPEG, MIDI
Sloj sesije	NetBIOS, SAP, SDP, NWLink
Transpotni sloj	TCP, UDP, SPX, RTP
Mrežni sloj	IP, IPX, ICMP, RIP, OSPF, BGP
Sloj linka za podatke	Ethernet, Token Ring, FDDI, ARP

Tabela 2.4.1. Pregled najčešće korišćenih protokola po OSI slojevima

Kako se podaci prenose kroz OSI model? Inicijalni prenos podataka u mreži započinje na aplikacionom sloju sistema koji šalje podatke. Podaci se prenose kroz svih sedam slojeva OSI modela sve dok ne stignu do fizičkog sloja na dnu OSI modela. Fizički sloj transmisionog sistema šalje podatke do prijemnog sistema. Prijemni sistem preuzima podatke sa svog fizičkog sloja i prosleđuje podatke kroz preostale slojeve prijemnog sistema sve dok podaci ne stignu do aplikacionog sloja na vrhu.

Servisi koje pružaju različiti protokoli bilo kog sloja OSI modela nisu redundantni. Na primer, ako protokol određenog sloja pruža određene servise, onda ni jedan drugi protokol bilo kog drugog sloja ne pruža iste servise. Protokoli koji pripadaju različitim slojevima možda imaju karakteristike sa sličnim ciljevima, ali se njihove funkcije u suštini razlikuju.

Protokoli odgovarajućih slojeva računara koji međusobno komuniciraju su komplementarni. Na primer, ako je protokol aplikacionog sloja računara koji šalje podatke zadužen za enkripciju podataka koji se prenose, od odgovarajućeg protokola aplikacionog sloja mašine koja prima podatke se očekuje da dekriptuje podatke koje primi.

Na slici 2.4.2. je ilustrovana komunikacija dva klijenta kroz OSI model. Podaci se prenose sa vrha do dna OSI modela jednog klijenta, a onda u obrnutom smeru od dna do vrha kada stigne do drugog klijenta.

Svaki sloj OSI modela može da komunicira samo sa slojevima koji se nalaze direktno ispod i iznad njega. Na primer, sloj 2 može da šalje i prima podatke samo od slojeva 1 i 3.



Slika 2.4.2. Protokoli rade na istom sloju OSI modela i kod predajnog i kod prijemnog sistema [1]

2.4.3. Enkapsulacija podataka

Protokoli različitih slojeva OSI modela komuniciraju uz pomoć enkapsulacije podataka. Svaki sloj u steku je odgovoran za dodavanje zaglavlja (*header*) i začelja (*footer*) – dodatnih bitova sa informacijama koje omogućavaju slojevima da komuniciraju i razmenjuju podatke. Na primer, kada transportni sloj primi podatke od sloja sesije, on dodaje svoje zaglavlje i začelje sa informacijama na podatke pre nego što ih prosledi sledećem sloju.

Proces enkapsulacije kreira jedinicu podataka - PDU (*Protocol Data Unit*), koja u sebi uključuje podatke koji se prenose i sva zaglavlja sa informacijama koja su dodata podacima. Dok se podaci prenose kroz OSI model, jedinica podataka se menja i raste kako se dodaju zaglavlja sa informacijama od strane različitih protokola. Jedinica podataka ima konačnu formu u trenutku kada stigne do fizičkog sloja, a odatle se šalje ka odredištu. Prijemni računar sklanja zaglavlja sa jedinice podataka dok se podaci prenose kroz OSI slojeve. U trenutku kada stigne do najvišeg sloja OSI modela, samo originalni podaci koji su i poslati ostaju u paketu.

Termin paket se odnosi na kompletnu jedinicu podataka PDU koja uključuje i zaglavlja sa informacijama od svih slojeva OSI modela.

Na slici 2.4.3. je ilustrovan primer kako enkapsulacija podataka izgleda u praksi, prilikom pokušaja otvaranja <u>http://www.google.com/</u> stranice. U ovom slučaju, moramo da generišemo paket sa zahtevom koji se prenosi od našeg računara sve do odredišnog servera. Ovaj scenario podrazumeva da je već uspostavljena TCP/IP komunikaciona sesija.

Sve kreće od aplikacionog sloja računara koji koristimo. Pokušavamo da pristupimo vebsajtu preko HTTP aplikacionog protokola. Podaci u našem paketu se prenose dole niz stek sve do transportnog sloja. TCP transportni protokol je zadužen za pouzdan prenos paketa i dodaje TCP zaglavlje. Nakon toga, TCP prosleđuje paket do mrežnog sloja. IP mrežni protokol kreira zaglavlje koje sadrži informacije o logičkom adresiranju i prosleđuje paket do Etherneta na sloju linka za podatke. Ethernet zaglavlje sadrži fizičke Ethernet adrese. Paket je sada u potpunosti sklopljen i može da se prosledi do fizičkog sloja, na kojem se prenosi kao niz 0 i 1 kroz mrežu.

Kompletiran paket prolazi kroz mrežni sistem kabliranja i na kraju dolazi do *Google* veb servera. Veb server započinje čitanje paketa od sloja linka za podatke. Nakon što je ta informacija obrađena, zaglavlje sloja linka za podatke se uklanja i prelazi se na procesiranje informacija

mrežnog sloja. Informacije o IP adresiranju su pročitane iz odgovarajućeg zaglavlja i nakon toga se zaglavlje uklanja i prelazi na transportni sloj. TCP informacije su pročitane kako bi se osiguralo da je paket stigao u nizu. Nakon toga se uklanja zaglavlje transportnog sloja, dok u paketu ostaju samo podaci aplikacionog sloja. Ti podaci se prosleđuju aplikaciji veb servera na kojem se hostuje veb sajt. Kao odgovor na paket koji je poslao klijent, server bi trebalo da pošalje TCP paket klijentu koji potvrđuje da je njegov zahtev primljen uz otvaranje odgovarajućeg *html* fajla.



Slika 2.4.3.Grafički prikaz enkapsulacije podataka između klijenta i servera [1]

Treba uzeti u obzir da nije svaki paket koji putuje mrežom generisan od strane protokola aplikacionog sloja, tako da se može naići na paktete koji sadrže samo informacije protokola koji pripadaju drugom, trećem ili četvrtom sloju.

2.4.4. Mrežni uređaji

Najčešće korišćeni mrežni uređaji su:

- Hub Hub predstavlja uređaj sa više RJ-45 portova. Broj portova zavisi od kapaciteta mreže u kojoj hub radi. Hub je uređaj koji radi na fizičkom sloju OSI modela. Uzima pakete koji su stigli sa jednog porta i prenosi (ponavlja) ih ka svakom portu na uređaju. Klijenti povezani na portove huba koji su primili odgovarajuće pakete proveravaju da li je paket namenjen njima na osnovu MAC (*Media Access Control*) adrese i primaju paket ako je namenjen njima. Svi ostali klijenti povezani na ostale portove odbacuju pakete koji nisu namenjeni njima. Hubovi se retko koriste u modernim ili gustim mrežama zato što generišu dosta nepotrebnog mrežnog saobraćaja i mogu da rade samo u poludupleksmodu (ne mogu istovremeno da šalju i da primaju podatke).
- Svič Kao i hub, svič je dizajniran tako da ponavlja pakete, ali za razliku od huba, ne šalje podatke ka svakom portu, nego samo ka onom portu kojem su podaci i namenjeni. Svič izgleda kao i hub, s tim što se sa većim svičevima upravlja putem specijalizovanog softvera ili veb interfejsa. Takav tip svičeva pruža više funkcija koje mogu biti korisne prilikom upravljanja mrežom (mogućnost da se određeni portovi otvore ili zatvore za mrežni saobraćaj, da se vide specifičnosti porta, da se promeni konfiguracija ili da se

odradi daljinski *reboot*). Svičevi čuvaju MAC adrese svih povezanih uređaja u CAM tabeli. Kada stigne paket, svič čita informacije zaglavlja drugog sloja i na osnovu CAM tabele određuje na koji port treba da pošalje paket. Svičevi šalju pakete samo ka određenim portovima i na taj način značajno smanjuje mrežni saobraćaj u odnosu na hub.

 Ruter –Ruter je napredan mrežni uređaj sa mnogo više funkcionalnosti u odnosu na svič ili hub. Ruteri rade na mrežnom sloju OSI modela i zaduženi su za prosleđivanje paketa između dve ili više mreža. Proces koji koriste ruteri prilikom usmeravanja saobraćaja između mreža naziva se rutiranje. Nekoliko tipova protokola rutiranja određuju kako se različiti tipovi paketa rutiraju ka drugim mrežama. Ruteri obično koriste IP adrese kako bi na jedinstven način identifikovali uređaje u mreži. Veličina i broj rutera u jednoj mreži zavisi od veličine i funkcije same mreže.

2.4.5. Klasifikacija saobraćaja

Mrežni saobraćaj se može podeliti u tri veće klase: brodkast,multikast i unikast. Svaka klasa ima različite karakteristike koje određuju kako mrežni uređaji rukuju sa paketima u toj klasi.

- Brodkastsaobraćaj –Brodkast paket se šalje ka svim portovima u mrežnom segmentu. Postoji brodkast saobraćaj i na sloju 2 i na sloju 3 OSI modela. Postoje rezervisane brodkast adrese i bilo koji saobraćaj koji se šalje ka tim adresama, šalje se ka svim portovima koji pripadaju tom mrežnom segmentu.
- Multikast saobraćaj –Multikast paket se simultano prenosi sa jednog izvora do više odredišnih tačaka u istom trenutku. Glavni cilj multikasta je korišćenje što manjeg protoka usled simultanog prenosa (uz što manji broj duplih paketa). Primarni metod implementacije multikasta je kroz šemu adresiranja koja grupiše prijemnike paketa u multikast grupu. Na taj način se sprečava da paket stigne do računara kojem nije namenjen.
- Unikastsaobraćaj Unikast paket se prenosi direktno od jednog do drugog računara. Detalji prenosa zavise od protokola koji se koristi.

3.PRIKUPLJANJE PAKETA

Ključna odluka kako bi analiziranje paketa bilo efikasno je gde treba pozicionirati aplikaciju za prikupljanje paketa. Ponekad je teže pravilno pozicionirati aplikaciju za prikupljanje paketa od samog analiziranja prikupljenih paketa.

3.1. Promiscuous mod

Pre nego što započnemo sam proces prikupljanja paketa, potrebna nam je mrežna kartica NIC (*Network Interface Card*) koja podržava *promiscuous* mod. *Promiscuous* mod dozvoljava mrežnoj kartici da vidi sve pakete koji se kreću kroz mrežu.

Uz brodkast mrežni saobraćaj, podrazumeva se da klijent dobije pakete koji nisu namenjeni direktno njemu. ARP protokol se koristi za određivanje MAC adrese koja odgovara određenoj IP adresi. Kako bi pronašao odgovarajuću MAC adresu, ARP protokol šalje brodkast paket ka svim uređajima u brodkast domenu u nadi da će odgovarajući klijent odogovoriti.

Brodkast domen (mrežni segment u kome računari mogu razmenjivati podatke direktno, bez korišćenja rutera) može da sadrži više računara, ali bi samo jedan klijent trebalo da bude zainteresovan za ARP brodkast paket. Bilo bi krajnje neefikasno kada bi svaki računar u mreži zaista procesirao ARP paket. Umesto toga, mrežne kartice uređaja koji se nalaze u datoj mreži analiziraju paket i odbacuju ga ako odrede da nije za njih. Na taj način, paket ne dolazi do procesora.

Odbacivanje paketa koji nisu namenjeni datom hostu poboljšava efikasnost samog procesa, ali nije dobro u smislu analiziranja paketa. Kao neko ko analizira pakete, obično želimo da vidimo sve pakete koji prolaze kroz mrežu, kako ne bismo bili u situaciji da izgubimo neku ključnu informaciju.

Prilikom rada u *promiscuous* modu, mrežna kartica prosleđuje svaki paket koji vidi procesoru, bez obzira na adresiranje. Kada paket stigne do procesora, aplikacija može da ga pokupi.

3.2. Prikupljanje paketa u hub okruženju

Prikupljanje paketa u mreži koja ima hub uređaje je idealno rešenje. Kao što znamo, hub prosleđuje sav saobraćaj na sve portove koji su povezani na njega. Samim tim, jedino što je potrebno za analiziranje saobraćaja računara koji je povezan na hub je povezivanje na bilo koji prazan port na tom hubu. Na taj način je moguće pratiti svu komunikaciju od/do određenog računara, kao i komunikaciju između bilo kojih uređaja koji su povezani na dati hub.

Na slici 3.2.1. je ilustrovan vidljivi prozor (*visibility window*) koji je neograničen u smislu mreže koja se zasniva na hub uređaju. Vidljivi prozor predstavlja skup uređaja jedne mreže čiji je saobraćaj vidljiv aplikaciji za prikupljanje i analizu paketa.



Slika 3.2.1. Vidljivi prozor mreže u hub okruženju [1]

Mreže koje se zasnivaju na hub uređaju su veoma retke u današnje vreme zbog nedostataka i loših strana koje nose sa sobom. Samo jedan uređaj može da šalje ili prima podatke u datom trenutku, uređaji koji su povezani na hub moraju da se takmiče međusobno kako bi koristili protok u mreži. Kada dva ili više uređaja komunicira u isto vreme, dolazi do kolizije paketa. Rezultat kolizije može biti gubitak paketa, a uređaj koji je slao pakete pokušava ponovo da ih pošalje čime se samo povećava mrežno zagušenje i učestanost kolizija. Zbog svega toga imamo znatno više mreža koje koriste svičeve.

3.3. Prikupljanje paketa u svič okruženju

Svičevi omogućavaju efikasan način za prenos podataka kroz brodkast, multikast i unikast saobraćaj. Kao dodatak, svičevi dozvoljavaju pun dupleks komunikaciju, što znači da uređaji istovremeno mogu i da primaju i da šalju podatke.

U smislu analiziranja paketa, svičevi dodaju novi nivo kompleksnosti. Ako priključimo aplikaciju za prikupljanje paketa na port sviča, možemo samo da vidimo brodkast saobraćaj i saobraćaj koji prima i šalje sam uređaj koji je povezan na port.

Na slici 3.3.1. je ilustrovan vidljivi prozor (visibility window) mreže u svič okruženju.



Slika 3.3.1. Vidljivi prozor mreže u svič okruženju[1]

Postoje četiri načina za snimanje saobraćaja u svič okruženju: *port mirroring, hubbing out, using a tap* i ARP *cache poisoning.*

3.3.1. Port Mirroring

Port Mirroring je verovatno najlakši način za prikupljanje saobraćaja u svič okruženju. U ovom slučaju morate imati pristup komandnoj liniji ili interfejsu upravljanja mrežom sviča na koji je povezan računar čiji saobraćaj želimo da snimimo. Pored toga, svič mora da da ima podršku za *port mirroring*, kao i prazan port na koji možete da se priključite.

Kako bismo omogućili *port mirroring*, moramo uneti komande na osnovu kojih svič kopira sav saobraćaj sa jednog na drugi port. Na primer, ako želimo da snimimo saobraćaj sa porta 3 određenog sviča, potrebno je da se poveže aplikacija za analizu na port 4, odraditi port mirroring ta dva porta i na taj način možete videti sav saobraćaj koji šalje ili prima uređaj koji je povezan na portu 3.

Na slici 3.3.2. je ilustrovan prošireni vidljivi prozor (*visibility window*) mreže u svič okruženju, uz korišćenje *port mirroring* tehnike.



Slika 3.3.2. Vidljivi prozor mreže u svič okruženju – tehnika port mirroring[1]

Način na koji podešavamo*port mirroring* zavisi od proizvođača sviča koji će se koristiti. Za većinu svičeva potrebno je logovanje na interfejs komandne linije i unošenje odgovarajućih komandi. Nekoliko najčešće korišćenih komandi je prikazano u tabeli 3.3.1.

PROIZVOĐAČ	KOMANDA
Cisco	set span <source port=""/> <destination port=""></destination>
Enterasys	set port mirroring create <source port=""/> <destination port=""></destination>
Nortel	port-mirroring mode mirror-port <source port=""/> monitor- port <destination port=""></destination>

Prilikom korišćenja ove tehnike, potrebno je obratiti pažnjuna protok portova koji se koriste za ovu analizu. Neki proizvođači svičeva dozvoljavaju "preslikavanje" više portova u jednom portu, što može biti korisno prilikom analiziranja komunikacije između dva ili više uređaja povezana na jedan svič. Za svič sa 24 porta i primenite *port mirroring* 23 pun dupleks 100Mb/s porta u jedan port, do tog porta potencijalno može doći 4600Mb/s. To je znatno iznad fizičkog praga jednog porta, što može dovesti do gubitka paketa i usporavanja same mreže kada mrežni saobraćaj dostigne određeni nivo. U ovim situacijama, svičevi mogu odbaciti višak paketa ili čak i da pauiziraju interni prenos što dovodi po onemogućavanja celokupne komunikacije.

3.3.2. Hubbing Out

Drugatehnikakoja može prikupiti saobraćaj u svič okruženju je *hubbing out*. Ova tehnika podrazumeva povezivanje uređaja koji želi da se analizira i samog sistema za analizu direktno na hub uređaj. Mnogi ljudi vide ovu tehniku kao neki vid varanja, ali ona zaista predstavlja odlično rešenje u situaciji kada *port mirroring* nije izvodljiv, ali i dalje imamo fizički pristup sviču na koji je povezan uređaj čiji saobraćaj želimo da analiziramo.

Sve što nam je potrebno za *hubbing out* je hub i nekoliko mrežnih kablova. Povezivanje se ostvaruje kroz niz sledećih koraka:

- 1) Potrebno je da se ciljniuređaj čiji saobraćaj želimo da analiziramo diskonektuje sa sviča
- 2) Nakon toga se ciljni uređaj povezuje mrežnim kablom na hub
- 3) Analizator saobraćaja takođe se povezuje drugim mrežnim kablom na hub
- Hub se povezuje na mrežu tako što se postavi treći mrežni kabl od huba do mrežnog sviča

Ovom tehnikom ciljni uređaj i analizator saobraćaja se nalaze u istom brodkast domenu i sav saobraćaj sa ciljnog uređaja će biti prenet na sve portove. Na taj način će saobraćaj sa ciljnog uređaja doći i do analizatora koji može da prikupi sve pakete, kao što je ilustrovano na slici 3.3.3.



Slika 3.3.3. Vidljivi prozor mreže u svič okruženju – tehnika hubbing out[1]

U većini situacija, tehnika *hubbing out* će redukovati dupleksciljnog uređaja sa punog dupleksa na poludupleks. Iako ova tehnika nije najbolji način za analiziranje saobraćaja, ponekad je to jedina opcija u situaciji kada svič ne podržava *port mirroring*. Bitno je napomenuti da hub zahteva i priključak za napajanje, koji se ponekad teško može pronaći.

Još jedan problem na koji možemo naići, ako želimo da koristimo ovu tehniku, je u tome što se hubovi više ne proizvode u tolikoj meri i izuzetno je teško naći hub u nekoj prodavnici tehničke opreme.

3.3.3. Using a tap

Mrežni *tap* je hardverski uređaj koji možete da postavite između dve tačke u sistemu kabliranja, kako biste prikupili pakete koji se razmenjuju između te dve tačke.

Primarno postoje dve vrste mrežnih *tap* uređaja: agregacioni (*aggregated*) i neagregacioni (*nonaggregated*). Glavna razlika između ova dva tipa je u broju portova. Agregacioni *tap* ima 3 porta, dok neagregacioni *tap* ima 4 porta.

Tap uređaji takođe zahtevaju priključak za napajanje, a neki u sebi sadrže i baterije za vremenski kratko prikupljanje paketa, bez potrebe priključenja na električnu utičnicu.

Agregacioni *tap* je najlakši za korišćenje. Ima samo jedan fizički port za snimanje bidirekcionog saobraćaja. Kako bi se prikupio sav saobraćaj od i do pojedinačnog računara koji je povezan na svič, potrebno je pratiti sledeće korake:

- 1) Diskonektovati računar sa sviča
- 2) Staviti jedan kraj mrežnog kabla u računar, a drugi kraj na *in* port na *tap* uređaju
- 3) Staviti jedan kraj drugog mrežnog kabla na *out* port na *tap* uređaju, a drugi kraj na mrežni svič
- 4) Staviti jedan kraj trećeg mrežnog kabla u *monitor* port na *tap* uređaju, a drugi kraj u računar na kojem se nalazi aplikacija za prikupljanje saobraćaja

Agregacioni *tap* uređaj bi trebalo da bude povezan kao što je prikazano na slici 3.3.4 U tom slučaju, aplikacija bi trebalo da prikuplja sav saobraćaj koji dolazi i odlazi od ciljnog računara koji je povezan na *tap* uređaj.



Slika 3.3.4.Korišćenje agregacionog tap uređaja za presretanje mrežnog saobraćaja [1]

Neagragacioni *tap* uređaj je nešto kompleksniji u odnosu na agregacioni uređaj, ali dozvoljava i više fleksibilnosti prilikom prikupljanja saobraćaja. Umesto samo jednog *monitor* porta koji može da se koristi za snimanje bidirekcione komunikacije, neagregacioni *tap* uređaj ima dva porta za monitoring. Jedan port se koristi za snimanje saobraćaja u jednom smeru (od računara koji je konektovan na *tap* uređaj), a drugi port se koristi za snimanje saobraćaja u drugom smeru (do računara koji je konektovan na *tap* uređaj).

Kako bi se prikupio sav saobraćaj od i do pojedinačnog računara koji je povezan na svič, potrebno je pratiti sledeće korake:

- 1) Diskonektovati računar sa sviča
- 2) Staviti jedan kraj mrežnog kabla u računar, a drugi kraj na *in* port na *tap* uređaju
- Staviti jedan kraj drugog mrežnog kabla na *out* port na *tap* uređaju, a drugi kraj na mrežni svič
- 4) Staviti jedan kraj trećeg mrežnog kabla u *monitor A* port na *tap* uređaju, a drugi kraj u jednu mrežnu karticu računara na kojem se nalazi aplikacija za prikupljanje saobraćaja
- 5) Staviti jedan kraj četvrtog mrežnog kabla u *monitor B* port na *tap* uređaju, a drugi kraj u drugu mrežnu karticu računara na kojem se nalazi aplikacija za prikupljanje saobraćaja

Neagregacioni tap uređaj bi trebalo da bude povezan kao što je prikazano na slici 3.3.5.



Slika 3.3.5.Korišćenje neagregacionog tap uređaja za presretanje mrežnog saobraćaja [1]

Ako uporedimo obe vrste *tap* uređaja, agregacioni uređaji su prioritetniji zato što zahtevaju manje kabliranja i nisu potrebne dve mrežne kartice na računaru koji se koristi za presretanje saobraćaja. Ipak, u situacijama kada se snima veliki obim saobraćaja ili ako je od interesa samo saobraćaj koji se odigrava u jednom smeru, više se koriste neagragacioni uređaji.

3.3.4. ARP cache poisoning

Svi uređaji u mreži komuniciraju međusobno preko IP adresa trećeg sloja OSI modela. Kako svičevi rade na drugom sloju OSI modela sa MAC adresama. Ako MAC adresa nije poznata, koristi se ARP protokol koji prevodi poznatu IP adresu u odgovarajuću MAC adresu.

Za računare povezane na Ethernet mrežu, ARP proces započinje kada jedan računarželi da komunicira sa drugim računarom. Računar koji šalje podatke prvo proverava u ARP keš memoriji da li već ima informaciju koja MAC adresa je povezana sa IP adresom računara kojem želi da šalje podatke. Ako nema pomenutu informaciju, šalje se ARP zahtev kroz brodkast paket koji stiže do svakog računara u Ethernet mreži. Kroz zahtev se postavlja pitanje koja MAC adresa odgovara IP adresi odredišnog računara.

Uređaji koji nemaju IP adresu koja se nalazi u ARP zahtevu, jednostavno odbacuju ARP zahtev. Odredišni računar šalje ARP odgovor u kojem se nalazi njegova MAC adresa. Izvorišni računar u tom trenutku dobija informaciju o MAC adresi koja mu je potrebna za dalju komunikaciju, a tu informaciju stavlja u svoju keš memoriju.

ARP cache poisoning je proces slanja ARP poruka Ethernet sviču ili ruteru sa lažnom MAC adresom kako bi se presreo saobraćaj drugog računara. Na slici 3.3.6. je ilustrovan ovaj proces. Napadač snima saobraćaj u svič okruženju. Nakon što presretne MAC adresu uređaja čiji saobraćaj želi da presretne, napadač šalje ARP pakete u kojima obaveštava uređaje u mreži da on ima tu MAC adresu i na taj način se menja sadržaj keš memorije, a svi paketi koji su bili namenjeni toj MAC adresi, završavaju kod napadača.



Slika 3.3.6.Korišćenje ARP cache poisoning tehnike za presretanje mrežnog saobraćaja [1]

ARP *cache poisoning* je napredna forma presretanja mrežnog saobraćaja u svič okruženju. Najčešće se koristi za napade u smislu slanja lažno adresiranih paketa ili kako bi se izazvao DoS (*Denial-of-Service*) napad. Ipak, ova tehnika predstavlja legitiman način za snimanje saobraćaja ciljnog uređaja. Za primenu ove tehnike može se koristiti popularna sigurnosna alatka *Cain & Abel*, kojoj je to samo jedna od funkcionalnosti koje nudi.

Prilikom korišćenja ove tehnike treba biti obazriv kako se ne bi koristila za presretanje saobraćaja na uređaju koji ima veći kapacitet link od linka sistema koji će vršiti analizu.

3.4. Pozicioniranje analizatora u svič okruženju

Do sada smo se susreli sa četiri različita načina za snimanje mrežnog saobraćaja u svič okruženju. Možemo dodati još jedan metod, ako uzmemo u obzir jednostavno instaliranje aplikacije na uređaju čiji saobraćaj želimo da snimimo – *direct install method*. U tabeli 3.4.1. je napravljen uporedni prikaz tehnika uz smernice kada bi trebalo koristiti svaku od njih.

TEHNIKA	SMERNICE
Port mirroring	 Obično se koristi zato što ne ostavlja nikakve tragove u mreži i ne generiše dodatne pakete Može da se konfiguriše tako da klijent i dalje bude online, što je korisno kada se tehnika primenjuje na ruteru ili portovima servera
Hubbing out	 Idealna u situacijama kada nije problem da host bude privremeno offline Neefikasna u situacijama kada je potrebno snimiti saobraćaj sa više hostova zato što će trenutno doći do kolizije i gubitka paketa Može doći do gubitka paketa modernih 100/1000Mb/s hostova zato što je većina hubova 10Mb/s
Using a tap	 Idealna u situacijama kada nije problem da host bude privremeno offline Jedina opcija za snimanje saobraćaja u optičkim mrežama Tap uređaji mogu da pariraju modernim mrežnim brzinama zbog čega su superiorniji od tehnike koja koristi hub Nije preporučljiva ako je budžet ograničen
ARP cache poisoning	 Posmatra se kao "prljava" tehnika zato što uključuje slanje paketa u mreži kako bi se saobraćaj prerutirao kroz analizator saobraćaja Može da bude efikasna tehnika kada je potrebno brzo snimanje saobraćaja uređaja dok radi online i kada port mirroring nije opcija
Direct install	 Obično se ne preporučuje zato što bilo koji problem sa hostom može dovesti do odbacivanja paketa ili manipulacije na takav način da se paketi ne prikazuju tačno Mrežna kartica hosta ne mora da radi u promiscuous modu Najbolja je za testno okruženje, ispitivanje osnovnih performansi i analiziranje prikupljenog saobraćaja na neki drugi način

Tabela 3.4.1.Analiza tehnika za snimanje saobraćaja koje se koriste u svič okruženju



Na slici 3.4.1. prikazan je dijagram koji može pomoći prilikom izbora tehnike snimanja saobraćaja.

Slika 3.4.1.Dijagram koji treba pratiti prilikom izbora tehnike snimanja saobraćaja [1]

Prilikom snimanja mrežnog saobraćaja trebalo bi da budemo što je više moguće nevidljivi. U savršenom svetu bi trebalo da skupljamo podatke koji su nam potrebni, a da ne ostavimo dokaze da smo bilo šta radili.

3.5. Prikupljanje paketa u ruter okruženju

Sve tehnike prikupljanja paketa koje se koriste u svič okruženju, dostupne su i u ruter okruženju. Potrebno je obratiti pažnju na važnost pozicioniranja alata za snimanje saobraćaja prilikom analize problema koji obuhvata više mrežnih segmenata.

U situacijama kada podaci prolaze kroz više rutera, važno je analizirati saobraćaj sa svih strana rutera. Ako kao primer uzmemo problem u komunikaciji između uređaja koji pripadaju mrežama D i A (ilustrovano na slici 3.5.1.). Ukoliko pratite saobraćaj uređaja u mreži D koji ima problem u komunikaciji sa uređajem u drugoj mreži, možete jasno videti podatke koji se prenose do drugog segmenta, ali ne možete videti podatke koje taj drugi segment dobija kao odgovor. U takvim slučajevima, potrebno je pomeriti analizator (mreža B) kako biste dobili potpunu sliku saobraćaja koji se prenosi između različitih segmenata. Tek tada se može otkriti razlog zbog kojeg dolazi do

prekida saobraćaja – da li su podaci odbačeni ili nisu pravilno rutirani od strane rutera koji pripada mreži B. Ponekad je potrebno analizirati saobraćaj na više uređaja u različitim mrežnim segmentima kako bi se otkrio problem.



Slika 3.5.1.Otkrivanje problema u komunikaciji između uređaja u različitim mrežama u ruter okruženju [1]

4. PROTOKOLI NIŽIH SLOJEVA OSI MODELA

4.1. ARP (Address Resolution Protocol)

Za komunikaciju u nekoj mreži koriste se i logičke i fizičke adrese. Korišćenje logičkih adresa omogućava komunikaciju između više mreža i indirektno povezanih uređaja. Korišćenje fizičkih adresa olakšava komunikaciju na mrežnom segmentu uređajima koji su direktno povezani preko sviča. U većini slučajeva, da bi došlo do komunikacije neophodno je da se koriste oba tipa adresiranja u isto vreme.

Svič koji povezuje uređaje u mreži koristi CAM (*Content Addressable Memory*) tabelu u kojoj se nalaze MAC adrese svih uređaja koji su povezani na svaki port tog sviča. Kada svič primi saobraćaj namenjen određenoj MAC adresi, na osnovu CAM tabele se vrši provera na koji port treba da se prosledi primljeni saobraćaj. Ako odredišna MAC adresa nije poznata, prenosni uređaj će prvo proveriti da li se MAC adresa nalazi u njegovoj keš memoriji. Ako ne pronađe MAC adresu ni na taj način, onda je neophodna dodatna komunikacija u samoj mreži.

U TCP/IP mrežama (sa IPv4) koristi se ARP protokol za preslikavanje IP adresa u odgovarajuće MAC adrese. ARP protokol je definisan u okviru RFC 826. Prilikom ARP procesa koriste se samo dva paketa – ARP zahtev (*request*) i ARP odgovor (*response*), kao što je ilustrovano na slici 4.1.1.



Slika 4.1.1.Ilustracija ARP procesa [1]

Brodkast paket sa ARP zahtevom (u kojem se pita za MAC adresu uređaja koji ima poznatu IP adresu) se šalje svakom uređaju na mrežnom segmentu. Svaki uređaj koji nema IP adresu navedenu u paketu sa ARP zahtevom jednostavno odbacuje paket. Uređaj koji ima odgovarajuću IP adresu šalje paket sa ARP odgovorom koji sadrži njegovu MAC adresu.

Na kraju ARP procesa, prenosni uređaj ažurira svoju keš memoriju i nakon toga može započeti prenos podataka.

Na slici 4.1.2. je prikazano ARP zaglavlje sa odgovarajućim poljima:

Address Resolution Protocol			
Ba Offset	0-7	8–15	
0	Hardwa	ие Туре	
16	Protoco	ы Туре	
32	Hardware Address Length	Protocol Address Length	
48	Oper	ation	
64	Sender Hardware Address [1st 16 Bits]		
80	Sender Hardware Address (2nd 16 Bits)		
96	Sender Hardware Address (3rd 16 Bits)		
112	Sender Protocol Address (1st 1 6 Bits)		
128	Sender Protocol Address (2nd 16 Bits)		
144	Target Hardware Address [1st 16 Bits]		
160	Target Hardware Address (2nd 16 Bits)		
176	Target Hardware Address (3rd 16 Bits)		
192	Target Protocol Address (1 st 16 Bits)		
208	Torget Protocol Address (2nd 16 Bits)		

Slika 4.1.2. Prikaz ARP zaglavlja [1]

Polja ARP zaglavlja su:

- Hardware Type-tip drugog sloja koji se koristi (u većini slučajeva Ethernet tip 1)
- Protocol Type protokol višeg sloja zbog koga se traži ARP zahtev
- Hardware Address Length dužina hardverske adrese koja se koristi u bajtovima (6 za Ethernet)
- Protocol Address Length dužina logičke adrese određenog protokola u bajtovima
- Operation-funkcija ARP paketa: 1 za zahtev ili 2 za odgovor
- Sender Hardware Address hardverska adresa pošiljaoca
- Sender Protocol Address adresa koja se koristi u protokolima višeg sloja pošiljaoca
- **Target Hardware Address** hardverska adresa ciljnog uređaja (predstavljena nulama u ARP zahtevu)
- Target Protocol Address adresa koja se koristi u protokolima višeg sloja ciljnog uređaja

U mnogim slučajevima može doći do promene IP adrese samog uređaja. U tim situacijama mapiranje IP-MAC adresa koje se nalazi u keš memoriji hosta može biti pogrešno. Kako zbog toga ne bi došlo do grešaka prilikom komunikacije, šalje se *gratuitous*ARP paket kroz mrežu sa novim mapiranjem IP-MAC adresa, kao što se vidi na slici 4.1.3.



Slika 4.1.3. Gratuitous ARP proces [1]

ARP zaglavlje je isto kao prilikom slanja ARP zahteva, s tom razlikom da su IP adresa pošiljaoca i primaoca iste. Kada uređaji u mreži prime *gratuitous* ARP paket, ažurira se ARP tabela

svakog od njih sa novim mapiranjem IP-MAC adresa. Obično se šalje *gratuitous* ARP paket prilikom promene IP adrese uređaja. Takođe, neki operativni sistemi šalju ARP paket prilikom samog startovanja.

4.2. IP (Internet Protocol)

Glavna namena protokola trećeg sloja OSI modela je omogućavanje komunikacije između mreža. MAC adrese se koriste za komunikaciju u samoj mreži na drugom sloju OSI modela. Na isti način, mrežni sloj je odgovoran za adrese koje se koriste u međumrežnoj komunikaciji. Nekoliko protokola je zaduženo za to, ali se najčešće koristi IP protokol. Verzija 4 IP protokola (IPv4) je definisana u RFC 791.

IPv4 protokol je zadužen za prenos podataka između uređaja, bez obzira na to gde se nalaze krajnje tačke komunikacije. Jednostavna mreža u kojoj su svi uređaji povezani preko huba ili sviča predstavlja LAN (*Local Area Network*) mrežu. Veza između dve LAN mreže može se uspostaviti preko rutera. Hiljadu LAN mreža povezanih preko hiljadu rutera čine kompleksnu mrežu. Internet sam po sebi predstavlja kolekciju ogromnog broja LAN mreža i rutera.

IPv4 adrese su 32-bitne adrese koje se koriste za jedinstvenu identifikaciju uređaja koji su povezani na mrežu. Zbog lakšeg pamćenja, IP adrese se predstavljaju u *dotted-quad* notaciji. Svaki set bita 1 i 0 koji čine jednu IP adresu se konvertuje u broj između 0 i 255 u formatu A.B.C.D, kao što je ilustrovano na slici 4.2.1.



192.168.0.1

Slika 4.2.1. Dotted-quad notacija IPv4 adrese [1]

IP adresa se sastoji iz dva dela: mrežne adrese i adrese hosta. Mrežna adresa identifikuje LAN mrežu na koju je povezan uređaj, dok adresa hosta identifikuje sam uređaj u toj mreži. Za indikaciju podele IP adrese na adresu mreže (podmreže) i adresu hosta koristi se tzv. maska (*network mask* ili *subnet mask*), u kojoj su biti koji predstavljaju adresu mreže postavljeni na vrednost "1", dok su biti koji označavaju adresu hosta postavljeni na vrednost "0" (videti sliku 4.2.2.).



Slika 4.2.2.Mrežna maska određuje mrežnu adresu i adresu hosta [1]

IP adrese i mrežne maske najčešće se ispisuju u CIDR (*Classless Inter-Domain Routing*) zapisu. Na ovaj način, IP adresa se zapisuje u punom formatu, a posle kose crte (/) sledi broj bita koji predstavlja mrežni deo IP adrese. Npr. IP adresa 10.10.1.22 i mrežna maska 255.255.0.0 u CIDR formatu bi bile zapisane na sledeći način 10.10.1.22/16.

4.2.1. Format IPv4 datagrama

Izvorišna i odredišna IP adresa predstavljaju ključne komponente IPv4 datagrama, ali to je samo deo informacija koje možete naći u IPv4 datagramu. IPv4 datagram se sastoji od zaglavlja i korisnog segmenta. U zaglavlju se prenose informacije relevantne za funkcionisanje IPv4 protokola, a sastoji se od obaveznog dela dužine 20bajtova i opcionog dela, koji može imati dužinu 4-40 bajtova. Na slici 4.2.3. prikazana je struktura IPv4 paketa.

	8		Interne	t Protocol	
Bit Offset	0-3	4-7	8-15	16-18	19-31
0	Version	HDR Length	Type of Service		Total Length
32	Identification		Flags	Fragment Offset	
64	Time to Live Protocol		6	Header Checksum	
96	Source IP Address				
128	Destination IP Address				
160	Options				
160 or 192+	Data				

Slika 4.2.3.Struktura IPv4 paketa [1]

Polja IP zaglavlja su:

- Version definiše verziju protokola (npr. 4 za IPv4)
- Header Length dužina IP zaglavlja izražena u 32-bitnim rečima
- Type of Service uloga u kvalitetu servisa
- Total Length ukupna dužina datagrama (zaglavlje + podaci) izražena u oktetima (max 65535)
- Identification jedinstveni identifikacioni broj koji dodeljuje predajnik pomoć pri rekonstrukcijioriginalnog datagrama iz fragmenata
- Flags kontroliše fragmentiranje, koristi se za identifikovanje da li je paket deo sekvence fragmentiranih paketa
- Fragment Offset pokazuje mesto fragmenta u datagramu
- **Time to Live** vreme života definiše maksimalan dozvoljeni broj hopova datagrama kroz mrežu
- **Protocol** identifikuje protokol višeg sloja
- Header Checksum 16-bitni kontrolni zbir (samo za zaglavlje) koji se koristi za proveru da li je sadržaj IP zaglavlja oštećen
- Source IP Address-32-bitna adresa kojom se identifikuje predajnik paketa
- Destination IP Address 32-bitna adresa kojom se identifikuje prijemnik paketa
- Options opcione informacije o kontroli mreže, debagovanju, rutiranju i merenjima
- Data stvarni podaci koji se prenose

4.2.2. TTL (Time to Live)

TTL vrednost definiše maksimalan vremenski period ili maksimalan broj hopova (kroz rutere) paketa kroz mrežu pre nego što se paket odbaci. TTL se definiše prilikom kreiranja samog paketa i u principu se broj hopova dekrementira za 1 svaki put kada se paket prosledi od strane rutera. Npr. ako paket ima TTL vrednost 2, prvi ruter do koga stiže paket smanjiće TTL na 1 i

proslediti ka drugom ruteru. Taj drugi ruter će smanjiti vrednost TTL na 0 i ako se kranje odredište paketa ne nalazi u datoj mreži rutera, paket će biti odbačen (videti sliku 4.2.4.).



Slika 4.2.4.TTL vrednost paketa se smanjuje za 1 svaki put kada ruter prosledi paket dalje [1]

TTL vrednost je veoma bitna stavka zaglavlja IP datagrama. Naravno da nam je bitno vreme koje je potrebno da paket stigne od predajnika do prijemnika. Međutim, ako uzmemo u obzir da paket mora da prođe kroz više rutera na putu do svog odredišta. U nekom trenutku na toj putanji, paket može doći do pogrešno konfigurisanog rutera i izgubiti putanju do krajnjeg odredišta. U takvim slučajevima može doći do prosleđivanja paketa okolo u mreži u beskonačnoj petlji.

Beskonačna petlja u kojoj paket luta kroz mrežu može izazvati dosta problema, uključujući smanjenje protoka i na kraju DoS (*Denial of Service*). Kako bi se taj problem izbegao, stvoreno je TTL polje u IP zaglavlju.

4.2.3. IP fragmentacija

Fragmentacija paketa je funkcija koja omogućava pouzdan prenos podataka kroz različite tipove mreža, na taj način što se paketi dele na manje fragmente.

Fragmentacija paketa se zasniva na MTU (*Maximum Transmission Unit*) veličini protokola drugog sloja koji se koristi i konfiguraciji uređaja koji koriste protokole drugog sloja OSI modela. U većini slučajeva, protokol drugog sloja koji se koristi je Ethernet. Ethernet ima *default* vrednost MTU 1500, što znači da je maksimalna veličina paketa koji može da se pošalje kroz Ethernet mrežu 1500 bajtova (ne uključujući u tu vrednost 14-bajtno Ethernet zaglavlje). Iako postoje standardna MTU podešavanja na uređajima, postoji mogućnost ručnog rekonfigurisanja MTU veličine u većini slučajeva, na Windows ili Linux sistemima, kao i interfejsima samih rutera.

Kada se uređaj priprema za slanje IP paketa, potrebno je da proveri da li mora da fragmentira paket, tako što se veličina paketa poredi sa MTU veličinom mrežnog interfejsa sa kog će paket biti poslat. Ako je veličina paketa veća od MTU, paket će biti podeljen na fragmente, kroz sledeće korake:

- 1) Uređaj deli podatke na broj paketa koji je potreban kako bi se podaci uspešno preneli
- 2) Polje Total Length svakog IP zaglavlja se setuje na veličinu segmenta svakog fragmenta
- 3) Polje *More Fragments flag* je setovano na 1 za sve pakete podataka koji se prenose, osim kod poslednjeg paketa
- 4) Polje Fragment Offset je setovano u IP zaglavlju fragmenata
- 5) Može da započne prenos paketa

4.3. TCP (*Transmission Control Protocol*)

TCP je konektivno orijentisan protokol, namenjen da obezbedi pouzdanu komunikaciju između parova procesa u hostovima koji se mogu nalaziti u različitim, ali međusobno povezanim

mrežama. TCP, koji je definisan u RFC 793, radi na četvrtom sloju OSI modela. Zadatak TCP protokola je da ispravi greške nastale u prenosu kroz mrežu, koje se mogu reflektovati gubitkom, oštećenjem, dupliranjem ili pogrešnim redosledom paketa. Dosta protokola aplikacionog sloja oslanja se na TCP i IP kako bi paketi stigli do kranjeg odredišta.

4.3.1. TCP zaglavlje

TCP pruža veliki broj funkcionalnosti koje se ogledaju u kompleksnosti TCP zaglavlja. Format TCP zaglavlja je prikazan na slici 4.3.1.

		Tra	smission Contr	ol Protocol
Bit Offset	0-3	4-7	8-15	16-31
0	Source Port Destination Port			
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset	Reserved	Flags	Window Size
128	Checksum Urgent Pointer			
160	Options			

Slika 4.3.1.TCP zaglavlje [1]

Polja TCP zaglavlja su:

- Source Port-broj izvornog porta koji šalje paket
- Destination Port-broj odredišnog porta koji prima paket
- Sequence Number-broj prvog "data" bajta u segmentu koji se koristi za identifikaciju TCP segmenta kako bi se osiguralo da nijedan deo podataka ne fali
- Acknowledgment Number-ako je setovan kontrolni bit ACK, ovo polje sadrži vrednost sledećeg rednog broja čiji se prijem očekuje
- Data Offset broj 32-bitnih reči u TCP zaglavlju (ukazuje gde počinju podaci, sa kojim ofsetom)
- **Reserved** rezervisano za buduću upotrebu, mora imati vrednost 0
- Flags-kontrolni biti: U(URG) ako je setovan, polje Urgent Pointer je od značaja; A(ACK) – ako je setovan, polje "Acknowledgment Number" je od značaja; P(PSH) – funkcija push; R(RST) – reset veze; S(SYN) – sinhronizacija rednih brojeva; F(FIN) – nema više podataka za slanje
- Window Size-označava veličinu prijemnog prozora
- Checksum- kontrolna suma koja osigurava da je kompletan sadržaj TCP zaglavlja i podataka stigao na odredište
- Urgent Pointer-ako je setovan kontrolni bit URG, pokazuje redni broj okteta iza koga slede hitni podaci
- Options multipli od 8 bita; npr. najveća veličina segmenta i dr.

4.3.2. TCP portovi

Celokupna TCP komunikacija se odigrava korišćenjem izvorišnog i odredišnog porta koji se mogu naći u svakom TCP zaglavlju.

Kako bi se podaci preneli određenoj aplikaciji na udaljenom serveru ili uređaju, TCP paket mora da zna port koji "osluškuje" udaljeni servis. Ako pokušate da pristupite aplikaciji preko porta koji nije konfigurisan za korišćenje, komunikacija će biti neuspešna.

Izvorišni port nije toliko bitan i može se odabrati nasumično. Udaljeni server će odrediti sa kojim portom treba da komunicira na osnovu originalnog paketa koji mu je poslat (videti sliku 4.3.2.).



Slika 4.3.2.TCP koristi portove za prenos podataka [1]

Postoji 65535 portova koji mogu da se koriste prilikom komunikacije preko TCP protokola. Za dodeljivanje brojeva portova ovlašćena je organizacija IANA (*Internet Assigned Numbers Authority*). Brojevi portova grupisani su u tri opsega:

- Dobro poznati portovi opseg od 0 do 1023, uz ignorisanje porta 0 koji je rezervisan
- **Registrovani portovi** opseg od 1024 do 49151
- Dinamički, privatni ili efemerni portovi opseg od 49152 do 65535

Dobro poznate brojeve portova koriste sistemski procesi koji pružaju široko rasprostranjene tipove mrežnih servisa. Na primer, FTP koristi port 20, SMTP koristi port 25, HTTP koristi port 80, BGP koristi port 179...

Registrovane brojeve portova dodeljuje IANA za specifične servise, pri čemu na većini sistema obični korisnici mogu da koriste brojeve portova iz ovog opsega.

Za dinamičke brojeve portova nije potrebna registracija: ovi brojevi se koriste za poseben privatne namene, privremeno ili za automatsko dodeljivanje efemernih (kratkoročnih) portova.

Povezivanjem broja porta sa adresom mrežnog sloja (IP) formira se tzv. *socket. Socket* predstavlja kranju tačku transportne veze i označava se uređenim parom IP_adresa:port.

Par *socket*-a jednoznačno identifikuje TCP konekciju. To znači da se isti par ne može koristiti za identifikaciju više od jedne TCP konekcije, kao i da se konekcija može identifikovati samo pomoću jednog para *socket*-a. Jedan *socket* se može istovremeno koristiti u različitim konekcijama.

4.3.3. TCP procedura "trostrukog rukovanja"

Nakon definisanja para *socket*-a koji predstavljaju kranje tačke TCP veze, u proceduri za uspostavu veze, TCP entiteti razmenjuju redne brojeve (polje *Sequence Number* u zaglavlju), da bi obezbedili ispravno potvrđivanje prijema podataka. Taj proces se naziva uspostavljanjem sinhronizacije, a izvršava se u tri koraka, zbog čega se procedura uspostave TCP veze naziva i procedurom "trostrukog rukovanja".

U prvom koraku, uređaj koji želi da komunicira (host A) šalje TCP paket ka uređaju sa kojim želi da komunicira (host B). Inicijalni paket ne sadrži podatke, a u TCP zaglavlju je setovan

SYN kontrolni bit, uključujući podatke o broju sekvence i MSS (*Maximum Segment Size*) veličini koji će da se koriste u procesu komunikacije. Host B odgovara na dobijeni TCP paket slanjem sličnog paketa sa setovanim SYN i ACK kontrolnim bitima, uz svoj inicijalni broj sekvence. Na kraju, host A šalje poslednji paket hostu B u kojem je samo setovan ACK kontrolni bit. Sam proces je ilustrovan na slici 4.3.3.



Slika 4.3.3.TCP procedura "trostrukog rukovanja" [1]

Nakon što se završi TCP procedura "trostrukog rukovanja", oba uređaja bi trebalo da imaju sve informacije koje su im potrebne kako bi uspešno razmenili podatke.

U nekim slučajevima, može se koristiti i TCP *teardown* koji podrazumeva slanje i četvrtog paketa u kojem je setovan FIN kontrolni bit kako bi se naznačio kraj veze (vidi sliku 4.3.4.).



Slika 4.3.4.TCP teardown procedura [1]

U idealnom svetu, svaka veza bi trebalo da se završi uz razmenu informacija o samom prekidu, uz TCP *teardown*proceduru. U stvarnosti, veze se često prekidaju iznenada. U tim slučajevima, koristi se TCP paket u kojem je setovan RST kontrolni bit. RST kontrolni bit se koristi kao indikator da je veza iznenada prekinuta ili kao odbijanje da se veza uopšte uspostavi.

Danas se od svih TCP implementacija zahteva da koriste algoritme kontrole zagušenja kao što su: spori početak, izbegavanje zagušenja, brza retransmisija i brzi oporavak.

4.4. UDP (User Datagram Protocol)

UDP je još jedan protokol četvrtog sloja OSI modela koji se obično koristi u modernim mrežama. Dok je TCP dizajniran za pouzdan prenos podataka sa kontrolom grešaka, namena UDP protokola je da pruži brzu razmenu podataka. UDP je protokol bez uspostave veze, bez formalnog uspostavljanja i raskidanja veze između dva hosta. UDP je definisan u RFC 768.

Protokoli koji se oslanjaju na UDP, kao aplikacioni protokoli DNS i DHCP kod kojih je veoma bitno da se paketi brzo šalju kroz mrežu, imaju sopstvenu kontrolu grešaka prilikom prenosa.

4.4.1. UDP zaglavlje

UDP zaglavlje je znatno manje i jednostavnije u poređenju sa TCP zaglavljem, kao što se može primetiti na slici 4.4.1.

User Datagram Protocol			
Bit 0–15 Offset		16-31	
0	Source Port	Destination Port	
32	Packet Length	Checksum	

Slika 4.4.1.UDP zaglavlje [1]

Polja UDP zaglavlja su:

- Source Port broj izvornog porta koji šalje paket
- **Destination Port** broj odredišnog porta koji prima paket
- Packet Length-dužina paketa izražena u bajtovima
- Checksum–opciona kontrolnasuma koja se koristi za detekciju greške na kompletnom UDP datagramu

UDP protokol je generalno namenjen za situacije u kojima je brzina isporuke podataka bitnija od integriteta, uz pretpostavku da će aplikacioni proces inicirati ponavljanje operacije u slučaju greške u prenosu.

4.5. ICMP (Internet Control Message Protocol)

ICMP je prateći kontrolni protokol IPv4, koji registruje greške detektovane u zaglavlju IPv4 datagrama, a koristi se i u dijagnostičke svrhe i za pomoćne procedure u procesu rutiranja. ICMP detektuje neregularnosti povezane sa formatom i/ili prosleđivanjem određenog IP datagrama i o tome obaveštava izvor datagrama (na osnovu adrese u zaglavlju).

4.5.1. ICMP zaglavlje

ICMP je deo IP protokola i oslanja se da će IP proslediti njegove poruke. ICMP ima relativno malo zaglavlje koje se menja u zavisnosti od svrhe samog zaglavlja. ICMP zaglavlje je prikazano na slici 4.5.1.

Internet Control Message Protocol					
Bit Offset	0-15		16-31		
0	Туре	Code	Checksum		
32	Variable				

Slika 4.5.1.ICMP zaglavlje [1]

Polja ICMP zaglavlja su:

- Type tip ili klasifikacija ICMP poruke, na osnovu RFC specifikacije
- Code podklasifikacija ICMP poruke, na osnovu RFC specifikacije
- Checksum-kontrolni zbir koji osigurava da je sadržaj ICMP zaglavlja i podataka stigao nepromenjen do svog odredišta
- Variable zavisi od polja *Type* i *Code*

U tabeli 4.5.1. dat je opis funkcija karakterističnih poruka ICMP protokola. Svaka poruka sadrži zaglavlje dužine 8 bajtova i telo (podatke) promenljive dužine. Poruke se direktno enkapsuliraju u IP datagrame, što znači da zaglavlje ICMP započinje odmah pre zaglavlja IPv4.

TIP ICMP PORUKE	OPIS
Destination unreachable	Paket nije mogao da bude isporučen
Time exceeded	Vrednost TTL je 0
Parameter problem	Neispravno zaglavlje
Source quench	Indikacija zagušenja
Redirect	Preusmeri paket
Echo request	Testiranje rutera
Echo reply	Odziv na echo request

Tabela 4.5.1. Tipovi ICMP poruka

Jedna od korisnih funkcija koje pruža ICMP protokol je tzv. "pingovanje". Pingovanje se koristi za testiranje veze između uređaja. Kako bi se koristila funkcija ping, potrebno je u komandnoj liniji uneti komandu ping <ip adresa>, u kojoj se unosi IP adresa uređaja koji se nalazi u mreži. Ako je ciljni uređaj u funkciji, ako izvorišni računar ima rutu do ciljnog uređaja i ako ne postoji *firewall* koji bi blokirao komunikaciju, trebalo bi da se vide odgovori na pingovanje.

Ukoliko je uspostavljena komunikacija, trebalo bi da se prikažu četiri uspešna odgovora koji prikazuju njihovu veličinu, RTT i TTL. Windows daje i informacije o ukupnom broju paketa koji je poslat, primljen i izgubljen. Ako je komunikacija neuspešna, u poruci se mogu videti razlozi neuspeha.

U suštini ping komanda šalje jedan paket u trenutku i osluškuje odgovor kako bi ustanovio da li je moguća komunikacija sa uređajem koji se pinguje (videti sliku 4.5.2.).



Slika 4.5.2.Ping komanda obuhvata dva koraka [1]

5. Protokoli viših slojeva osi modela

5.1. DHCP (Dynamic Host Configuration Protocol)

DHCP je protokol aplikacionog sloja OSI modela, koji omogućuje automatsko dodeljivanje IP adrese hostovima. DHCP server upravlja skupom IP adresa i parametrima konfiguracije klijenata, kao što su ime domena, serveri imena i drugi serveri.

Postoje tri metode dodeljivanja IP adresa:

- Dinamičko dodeljivanje IP adresa Administrator mreže dodeljuje opseg IP adresa DHCP serveru. Svaki klijent u LAN mreži konfigurisan je tako da traži IP adresu od servera, u fazi inicijalizacije. Taj proces "zahtev-odobrenje" funkcionipe po principu dodeljivanja adresa na određeno vreme. Posle isteka tog vremena, vrši se obnavljanje, kada klijent zadržava adresu za sledeći grant period ili mu se dodeljuje druga IP adresa.
- Automatsko dodeljivanje IP adresa Postupak je sličan dinamičkom dodeljivanju adresa, s tom razlikom što DHCP server održava tabelu dodeljenih IP adresa. Kada istekne grant period, prvi izbor DHCP servera je da klijentu ponovo dodeli istu IP adresu.
- Statičkododeljivanje IP adresa DHCP održava tabelu sa parovima IP adresa/MAC adresa. Tu tabelu manuelno popunjava administrator mreže. IP adrese se dodeljuju samo registrovanim klijentima, odnosno samo klijentima čije se MAC adrese nalaze u pomenutoj tabeli.

5.1.1. Struktura DHCP paketa

DHCP paket može da prenese velik broj informacija klijentu. Struktura DHCP paketa je prikazana na slici 5.1.1.

1	Dyne	amic Host Config	uration Protocol		
Bit Offset	(-15	16-31	31	
0	OpCode	Hardware Type	Hardware Length	Hops	
32	Transaction ID				
64	Seconds Elapsed		Flags		
96	Client IP Address				
128	Your IP Address				
160	Server IP Address				
196	Gateway IP Address				
228+	Client Hardware Address (16 bytes)				
		Server Host N	lame (64 bytes)		
		Boot File	(128 bytes)		
10	Options				

Slika 5.1.1.Struktura DHCP paketa [1]

Polja DHCP paketa su:

- **OpCode** pokazuje da li je paket DHCP zahtev ili odgovor
- Hardware Type tip hardverske adrese
- Hardware Length dužina hardverske adrese
- Hops koristi se za pronalaženje DHCP servera
- Transaction ID nasumičan broj koji povezuje zahtev i odgovor
- Seconds Elapsed sekunde od kada je klijent prvi put tražio adresu od DHCP servera
- Flags-tip saobraćaja koji DHCP klijent može da prihvati (unikast, brodkast)
- Client IP Address IP adresa klijenta
- Your IP Address IP adresa koju nudi DHCP server
- Server IP Address IP adresa DHCP servera
- Gateway IP Address IP adresa mrežnog default gateway-a
- Client Hardware Address MAC adresa klijenta
- Server Host Name host ime servera (opciono)
- **Boot File** boot fajl koji koristi DHCP (opciono)
- Options koristi se za dodatne funkcije DHCP paketa

5.1.2. DHCP proces obnavljanja

Primarni cilj DHCP protokola je dodeljivanje IP adrese klijentu u procesu obnavljanja. Proces obnavljanja se odigrava izmeu klijenta i DHCP servera, koristi četiri tipa DHCP paketa: *discover, offer, request* i *acknowledgment* (odatle i naziv DORA proces), kao što je ilustrovano na slici 5.1.2.



Slika 5.1.2.DHCP DORA (discover, offer, request i acknowledgment) proces [1]

U procesu inicijalizacije, klijent mora da locira raspoložive DHCP servere, uz slanje brodkast*discover*paket. DHCP server šalje *offer* paket kako bi ponudio svoje servise klijentu, tako što mu šalje informacije o samom servervu i adresiranju koje želi da pruži klijentu. Kada klijent primi *offer* paket od DHCP servera, on prihvata ponudu uz slanje *request* paketa. Na kraju, DHCP server šalje traženu IP adresu klijentu u okviru *acknowledgment* paketa i snima tu informaciju u svoju bazu podataka. Klijent nakon DORA procesa ima IP adresu i može da započne komunikaciju u mreži.

DHCP protokol se oslanja na UDP transportni protokol zato što je veoma bitno da klijent dobije što brže informaciju koju je tražio.

Kada DHCP server dodeli IP adresu uređaju, on je u stvari iznajmljuje na određeno vreme. Kada to vreme istekne, ponovo se odigrava DORA proces tokom koga se ponovo zahteva IP adresa od DHCP servera.

5.1.3. DHCP tipovi poruka

Jedina opcija koja mora biti prisutna u svim DHCP paketima je tip poruke koja se šalje. Postoji osam tipova poruka, koji su prikazani u tabeli 5.1.1.

TIP DHCP PORUKE	OPIS	
Discover	Šalje klijent kako bi locirao dostupne DHCP servere	
Offer	Šalje server klijentu kao odgovor na discover paket	
Request	Šalje klijent kako bi tražio ponuđene parametre od servera	
Decline	Šalje klijent serveru kako bi ukazao na nevažeće parametre u okviru paketa	
ACK	Šalje server klijentu sa traženim konfiguracionim parametrima	
NAK	Šalje klijent serveru kako bi odbio zahtev za konfiguracione parametre	
Release	Šalje klijent serveru kako bi otkazao zakup konfiguracionih parametara	
Inform	Šalje klijent serveru kako bi saznao konfiguracione parametre kada mu je već dodeljena IP adresa	

Tabela 5.1.1. Tipovi DHCP poruka

5.2. DNS (Domain Name System)

ASCII imena hostova i servera u Internetu uvedena su sa ciljem razdvajanja naziva uređaja od mrežnih (IP) adresa. Numeričke adrese su komplikovane i teško se pamte, a ako bi se uređaji identifikovali samo mrežnim adresama dodatno bi se komplikovala bilo koja promena mašine. Kako mreža radi samo sa numeričkim adresama, neophodan je mehanizam koji preslikava ASCII imena u mrežne adrese.

Sistem imena domena ili DNS predstavlja hijerarhijski organizovan sistem distribuiranih baza podataka koji implementira šemu imena domena i vrši preslikavanje imena u IP adrese.

5.2.1. Struktura DNS paketa

Struktura DNS paketa je prikazan na slici 5.2.1.

Domain Name System						
Bit Offset	0-15	16-31				
0	DNS ID Number	Q OpCode A T R R Z RCode				
32	Question Count	Answer Count				
64	Name Server Count	Additional Records Count				
96	Questions Section	Answers Section				
128	Authority Section	Additional Information Section				

Slika 5.2.1.Struktura DNS paketa [1]

Polja DNS paketa su:

- DNS ID Number broj koji povezuje DNS upit i DNS odgovor
- Query/Response (QR) označava da li je paket DNS upit ili DNS odgovor
- **OpCode** definiše tip upita koji se nalazi u poruci
- Authoritative Answers (AA) ako je AA vrednost setovana u paketu sa odgovorom, to znači da je odgovor stigao od DNS servera koji ima autoritet u okviru domena
- **Truncation (TC)** ukazuje da je odgovor nije potpun zato što je bio prevelik da stane u jedan paket
- **Recursion Desired (RD)** ako je RD vrednost setovana u paketu sa upitom, to znači da DNS klijent zahteva rekurzivan upit ako ciljni DNS server ne sadrži traženu informaciju
- **Recursion Available (RA)** ako je RA vrednost setovana u paketu sa odgovorom, to znači da DNS server podržava rekurzivne upite
- **Reserved (Z)**–rezervisano polje, u okviru RFC 1035 je definisano kao niz nula, ali se ponekad može koristiti kao ekstenzija Rcode polja
- **Response Code (RCode)** koristi se u DNS odgovoru kako bi se označilo prisustvo bilo koje greške
- Question Count broj ulaza u *question* delu
- Answer Count broj ulaza u *answer* delu
- Name Server Count broj zapisa o resursima DNS servera u okviru domena
- Additional Records Count broj drugih zapisa o resursima u dodatnim delovima
- Questions section polje promenljive veličine, sadrži jedan ili više upita koji će biti poslati DNS serveru
- Answers section polje promenljive veličine, sadrži jedan ili više zapisa o resursima koji odgovaraju na upite
- Authority section polje promenljive veličine koje sadrži zapise o resursima koji ukazuju na glavni DNS server koji može da se koristi u procesu rezolucije
- Additional Information section polje promenljive veličine koje sadrži zapise o resursima sa dodatnim informacijama vezanim za upit

Sa aspekta aplikacije, osnovna komponenta DNS sistema je procedura koja se naziva *Resolver*. Korisnički program poziva proceduru iz odgovarajuće biblioteke i definiše ime domena kao ulazni parametar. Procedura *Resolver* zatim šalje UDP paket lokalnom DNS serveru. Kada DNS server pronađe ime, vraća pridruženu IP adresu.

Svaki domen održava skup zapisa o resursima (*Resource Record*, RR). Zapis o resursima sastoji se od sledećih pet binarno kodiranih parametara: ime domena, vreme života, klasa, tip i vrednost. Za svaki domen tipično postoji veći broj zapisa. Vreme života (TTL) pokazuje koliko je zapis stabilan. Pojedini tipovi zapisa i pridružene vrednosti prikazani su u tabeli 5.2.1.

TIP DNS ZAPISA	OPIS	
А	IPv4 adresa hosta	
NS	Server za ovaj domen	
CNAME	Ime domena	
MX	Server za razmenu elektronske pošte	
TXT	Proizvoljan ASCII tekst	
AAAA	IPv6 adresa hosta	
IXFR	Postepena zona transfera	
AXFR	Puna zona transfera	

Tabela 5.2.1. Tipovi DNS zapisa

Prostor sa imenima DNS-a podeljen je na zone izmeđ kojih nema preklapanja. Svaka zona obuhvata deo DNS stabla i DNS servere koji poseduju informacije o toj zoni. Jedan DNS server je primarni, a može postojati jedan ili više sekundarnih DNS servera. procedura *Resolver* uvek se obraća lokalnom DNS serveru, koji po potrebi kontaktira udaljeni server. U opštem slučaju, moguće je da se zahtev procedure *Resolver* prosleđuje u više koraka, odnosno kroz nekoliko DNS servera, kao što je ilustrovano na slici 5.2.2.



5.3. HTTP (Hypertext Transfer Protocol)

HTTP je protokol koji se koristi za pristup podacima na WWW (*World Wide Web*), odnosno za komunikaciju veb pretraživača i servera. Najrasprostranjenija je verzija 1.1 ovog protokola, koja je definisana u RFC 2616. HTTP je protokol tipa upit-odziv, koji pretpostavlja da klijent (veb pretraživač) generiše upite za veb server. Server, koji čuva određene podatke ili obezbeđuje resurse (kao što su HTML fajlovi) ili izvršava neke druge funkcije, vraća odziv klijentu.



Slika 5.3.1.Razmena HTTP upita i odziva

HTTP klijent aktivira zahtev tako što uspostavlja TCP vezu sa serverom (koji zatim čeka na portu 80). Kada primi zahtev od klijenta, HTTP server vraća odziv. HTTP sesija je niz transakcija tipa zahtev-odziv. Linija upita u zahtevu sadrži podatke o tipu upita, URL i verziji HTTP. Linija statusa u odzivu sadrži podatke o verziji HTTP i statusu (rezultatu izvršavanja zahteva).

HTTP definiše metode, koje identifikuju traženu akciju (tabela 5.3.1.).
Tabela 5.3.1. Metode HTTP

NAZIV METODA	OPIS
OPTIONS	Vraća listu HTTP metoda koje podržava server, za definisani URL
GET	Zahteva prezentaciju definisanog resursa (ne sme se preduzimati nijedna druga akcija osim pretraživanja)
HEAD	Slično metodi GET, ali odziv ne sadrži telo poruke. Ovaj metod je koristan za pretraživanje meta-informacija u zaglavljima
POST	Šalje podatke koje definisani resurs treba da procesira
PUT	Ažurira prezentaciju definisanog resursa
DELETE	Briše definisani resurs
TRACE	Eho primljenog zahteva, tako da klijent ima uvid u eventualne promene koje su izvršili tranzitni serveri
CONNECT	Konvertuje traženu vezu u transparentan TCP/IP tunel; najčešće za potrebe realizacije bezbednosnih mehanizama

HTTPS (*HTTP Secure*) je HTTP protokol dopunjen bezbednosnim mehanizmima, koji omogućavaju poverljivost podataka i identifikaciju web servera. U URL šemi se identifikuje nizom karaktera *https*, a koristi TCP servise na portu 443. Protokol se takođe koristi za autentifikaciju klijenta, sa ciljem da se pristup serveru dozvoli samo ovlašćenim korisnicima.

6.PROGRAMSKI ALAT WIRESHARK

Programski alat Wireshark ima veoma bogatu istoriju. Originalno ga je razvio Džerald Kombs, diplomirani inženjer Univerziteta Misuri u Kanzas Sitiju. Prva verzija aplikacije, pod nazivom *Ethereal*, puštena je na tržište 1998. godine, uz GPL (*GNU Public License*) licencu. Osam godina kasnije, Kombs je promenio posao, a bivši poslodavac je zadržao sva prava na *Ethereal* ime i zaštitni znak. Zbog nemogućnosti korišćenja originalnog imena aplikacije, Kombs i ostatak razvojnog tima su napravili rebrending projekta pod nazivom Wireshark sredinom 2006. godine.

Wireshark je postao izuzetno popularan alat i njegov razvojni tim se sastoji od preko 500 saradnika. Program koji je postojao pod nazivom *Ethereal* se više ne razvija.

Prednosti korišćenja Wireshark aplikacije:

- **Podržani protokoli**–Wireshark podržava veliku većinu protokola, preko 850 protokola, od najčešćih protokola kao što su IP i DHCP do drugih protokola kao što su AppleTalk i BitTorrent. Kako je Wireshark *open source* model, podrška za nove protokole se dodaje prilikom svake naprednije i novije verzije aplikacije.
- User-friendliness-Wireshark interfejs je jedan od najjednostavnijih za razumevanje u
 poređenju sa drugim aplikacijama tog tipa. Ima grafički korisnički interfejs sa veoma
 jasnim menijem i opcijama koje nudi. Pored toga, pruža i nekoliko funkcija koje
 poboljšavaju korišćenje same aplikacije, kao što su razdvajanje protokola na osnovu boja
 i detaljnu grafičku prezentaciju prikupljenih paketa. Wireshark mogu koristiti i početnici,
 pored naprednijih korisnika.
- Cena –Wireshark je *open source*, besplatni softver, pod GPL licencom. Možete skinuti instalaciju za Wireshark potpuno besplatno i koristiti aplikaciju bilo u privatne ili komercijalne svrhe.
- Podrška internet zajednice–Kada se radi o besplatno distribuiranim softverima kao što je Wireshark, možda ne postoji formalna podrška, zobg čega se najčešće oslanjamo na korisničku bazu u smislu podrške. Na sreću, Wireshark zajednica je jedna od najaktivnijih u oblasti *open source* projekata. Wireshark veb stranica ima direktne linkove koji vode do nekoliko različitih oblika podrške, kao što su onlajn dokumentacija, wiki (*support* i *development*), FAQ (Frequently Asked Questions) najčešće postavljanja pitanja, mejling lista za dobijanje najnovijih informacija.
- **Podržani operativni sistemi**–Wireshark podržava većinu modernih operativnih sistema, uključujući Windows, Mac OS X i Linux platforme. Kompletna lista podržanih operativnih sistema može se naći na zvaničnoj veb stranici Wireshark aplikacije.

Sama instalacija Wireshark aplikacije je izuzetno jednostavna. Ipak, pre nego što krenemo sa samom instalacijom softvera, potrebno je proveriti da li sistem koji koristimo zadovoljava sledeće zahteve: 400MHz i brži procesori, 128MB RAM memorije, najmanje 75MB dostupnog prostora za samu instalaciju, NIC kartica koja podržava *promiscuous mode* i WinPcap drajver za prikupljanje paketa.

WinPcap drajver za prikupljanje paketa je Windows implementacija pcap aplikativnog programskog interfejsa (API, *Application Programming Interface*). Jednostavno rečeno, ovaj drajver u interakciji sa operativnim sistemom snima pakete podataka, prilagođava filtre i menja mod rada NIC kartice.

WinPcap možete instalirati i odvojeno, ali je preporučljivo da se instalira verzija koja dolazi uz Wireshark instalaciju zato što ste za nju sigurni da je testirana i da radi sa poslednjom verzijom aplikacije.

6.1. Osnovne opcije Wireshark programskog alata

Za analiziranje paketa i mrežnog saobraćaja korišćena je poslednja verzija Wireshark aplikacije 2.0.5. Osnovne informacije o samoj aplikaciji mogu se pronaći u padajućem meniju Help->About Wireshark (slika 6.1.1).

📕 About Wireshark	? <mark>×</mark>
Wireshark Authors Folders Plugins Keyboard Shortcuts License	
WIRESHARK	
Network Protocol Analyzer	
Version 2.0.5 (v2.0.5-0-ga3be9c6 from master-2.0)	
Copyright 1998-2016 Gerald Combs <gerald@wireshark.org> and contributors. License GPLv2+: GNU GPL version 2 or later <http: gpl-2.0.html="" licenses="" old-licenses="" www.gnu.org=""> This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.</http:></gerald@wireshark.org>	
Compiled (64-bit) with Qt 5.3.2, with WinPcap (4_1_3), with libz 1.2.8, with GLib 2.42.0, with SMI 0.4.8, with c-ares 1.11.0, with Lua 5.2, with GnuTLS 3.2.15, with Grypt 1.6.2, with MIT Kerberos, with GeoIP, with QtMultimedia, with AirPcap.	
Running on 64-bit Windows 7 Service Pack 1, build 7601, with locale Serbian (Latin)_Serbia.1250, with WinPcap version 4.1.3 (packet.dll version 4.1.0.2980), based on libpcap version 1.0 branch 1_0_rel0b (20091008), with GnuTLS 3.2.15, with Grypt 1.6.2, without AirPcap. Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz (with SSE4.2), with 8112MB of physical memory.	
Built using Microsoft Visual C++ 12.0 build 40629	
Wireshark is Open Source Software released under the GNU General Public License.	
Check the man page and http://www.wireshark.org for more information.	
	ОК

Slika 6.1.1.Osnovni podaci o Wireshark aplikaciji koja se koristi za analizu

Otvaranjem same aplikacije, nudi se opcija za izbor interfejsa na kojem želimo da snimamo pakete, kod *Capture* opcije koja se nalazi na samoj sredini prozora. Ponuda interfejsa koji mogu da se koriste za snimanje paketa, zavisi od uređaja koji koristimo, kao i mreža na koje je dati uređaj povezan. Možemo da analiziramo saobraćaj u LAN mreži, Wireless mreži, Bluetooth mreži itd. Pored opcije Capture, nude se još dve opcije: opcija Open – gde su izlistana poslednja tri pcap falja koja su otvarana i opcija Learn – ispod koje se može naći linkovi koji vode na sajtove:

<u>https://www.wireshark.org/docs</u> za korisničko uputstvo, <u>https://wiki.wireshark.org/</u> za wiki deo Wireshark sajta na kojem možemo naći korisne informacije, <u>https://ask.wireshark.org/</u> za deo sa pitanjima i odgovorima u okviru samog sajta i <u>https://www.wireshark.org/lists/</u> link koji vodi ka prijavi za mejling listu.

Snimanje samih paketa možemo pokrenuti iz menija ako kliknemo na Capture->Options (slika 6.1.2.).



Slika 6.1.2.Kartica Capture u meniju

Dobijamo prozor prikazan na slici 6.1.3.



Interface	Traffic	Link-layer Header	Promiscuous	Snaplen (B)	Buffer (MB)	Capture Filte	r
Vireless Network Connection	wh	Ethernet	enabled	default	2		
Bluetooth Network Connection	n	Ethernet	enabled	default	2		
17 Fashla provincijoju poda na sli bola						Max	anno Totarfan
Enable promiscuous mode on all int	erraces					Mar	age interrace
Tank on Alter for adapted interference	Enter a capture filter					*	Compile Bl

Slika 6.1.3.Kartica Input u okviru Capture Interface opcija

Kartica Input nam prikazuje informaciju o dostupnim interfejsima i da li je aktiviran *promiscuous* mod i na kojim interfejsima. U koloni Traffic možemo da vidimo i koji interfejs je trenutno aktivan (u primeru na slici 6.1.3 aktivan je interfejs bežične mreže – Wireless Network Connection).

Kartica Output nam nudi mogućnost izbora formata snimljenog saobraćaja (pcap-ng ili pcap), automatsko snimanje novog fajla nakon svakih xx kB ili xx sekundi (što može biti korisno ako želimo da ograničimo veličinu snimljenog fajla –npr. maksimalno 20MB kako bismo mogli da pošaljemo fajl mejlom). Kartica Output je prikazana na slici 6.1.4.

Karica Options nam pruža izbor displej opcija prilikom samog snimanja paketa: da li želimo da se lista paketa ažurira u realnom vremenu – dok traje snimanje, da li želimo da se lista paketa automatski skroluje tokom snimanja i da li želimo da se prikažu dodatne informacije prilikom snimanja. Postoji mogućnost izbora i za koje adrese želimo da se prikaže ime hosta ili uređaja. U ponudi je rezolucija MAC adresa, IP adresa i transportnih imena. Pored toga, u kartici Options možemo definisati kada želimo da zaustavimo sam proces snimanja paketa – npr. nakon određenog broja paketa/fajlova, kB ili sekundi. Kartica Options je prikazana na slici 6.1.5.

The Wireshark Network Analyzer	
Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help	
■ ⊿ 🔍 🕌 🖄 🖄 🤇 4 ↔ ↔ 쭆 🐨 🖢 🚍 🚍 9. 9. 9. 9. 표	
apply a display filter <ctrl-></ctrl->	
Wireshark - Capture Interfaces Input Output Options Cepture to a permanent file File: Leave blank to use a temporary file Output format: © pcoring © pcop © Greate a new file automatically after I Be Beboytes = Use a ring buffer with 2 B files	Browse
	Cose Trep
1	

You are running Wireshark 2.0.5 (v2.0.5-0-ga3be9c6 from master-2.0). You receive automatic updates

File





You are running Wireshark 2.0.5 (v2.0.5-0-ga3be9c6 from master-2.0). You receive automatic u

Slika 6.1.5.Kartica Options u okviru Capture Interface opcija

Nakon što izaberemo interfejs, način i vreme koje želimo da snimamo, možemo pokrenuti i samo snimanje paketa klikom na Capture->Start ili klikom na ikonicu za snimanje (start) odmah ispod File menija (slika 6.1.6.). Snimanje možemo pokrenuti i klikom na interfejs na kojem želimo da snimamo saobraćaj.

🛋 *V	/ireles	s Netwo	ork Cor	nnect	ion		
File	Edit	View	Go	Сар	ture Analyze	Statistics	Teleph
	đ	۲	010	۲	Options	Ct	rl+K
Ар	ply a d	isplay filt	ter <		Start	Ct	rl+E
No.	-	Time			Stop	Ct	rl+E
	1 (0.0000	00	a	Restart	Ct	rl+R 2
	2 (0.0002	34		Capture Filters.		2
	3 (0.4203	33		Refrech Interfac	or ES	
	4	3.0102	27		Nerrestratitenac	.es 10	2

Slika 6.1.6.Pokretanje snimanja paketa u wireshark aplikaciji klikom na start ikonicu

Ako unapred nismo definisali trenutak u kojem će snimanje paketa da se zaustavi, onda možemo da zaustavimo snimanje u bilo kom trenutku klikom na ikonicu stop ili klikom na Capture->Stop.

Snimljene pakete možemo sačuvati u odgovarajućem formatu za dalju analizu. Takođe, možemo snimiti saobraćaj na jednom uređaju, a analizu snimljenih paketa raditi na drugom uređaju.

U glavnom prozoru Wireshark aplikacije možemo videti sve pakete koji su snimljeni, uz detalje o svakom od njih (slika 6.1.7.).

Proba 08092016.pcap						E
File Edit View Go	Capture Analyze Statist	tics Telephony Wireless Tools Help				
4 🔳 🧟 🛞 🗋	े 🕱 🔄 🭳 🗢 😤	1 🖢 🚍 🚍 🍳 🍳 🔍 🛄				
Apply a display filter	<ctrl-></ctrl->					 •
No. Time	Source	Source port Destination	Destination port	Length Protocol	Info	Expert
1 0.000000	Micro-St_6b:c5:de	Broadcast		60 ARP	Who has 10.0.32.119? Tell 10.0.33.36	
2 0.024374	HuaweiTe_56:73:96	Spanning-tree-(for		119 STP	MST. Root = 32768/0/30:d1:7e:b2:da:77 Cost = 10000 Port = 0x8019	
3 1.464916	10.0.33.243	40505 239.255.255.250	1900	213 SSDP	M-SEARCH * HTTP/1.1	Chat
4 1.504581	10.0.32.219	60845 239.255.255.250	1900	216 SSDP	M-SEARCH * HTTP/1.1	Chat
5 2.174608	HuaweiTe_56:73:96	Spanning-tree-(for		119 STP	MST. Root = 32768/0/30:d1:7e:b2:da:77 Cost = 10000 Port = 0x8019	
6 2.412844	AsustekC_9a:d0:88	Broadcast		42 ARP	Who has 10.0.32.119? Tell 10.0.33.129	
7 2.466826	10.0.33.243	40505 239.255.255.250	1900	213 SSDP	M-SEARCH * HTTP/1.1	Chat
8 2.513435	10.0.32.219	60845 239.255.255.250	1900	216 SSDP	M-SEARCH * HTTP/1.1	Chat
9 3.002517	10.0.33.129	64232 192.168.2.17	161	120 SNMP	get-request 1.3.6.1.2.1.25.3.2.1.5.1 1.3.6.1.2.1.25.3.5.1.1.1 1.3.6.	
10 3.388159	AsustekC 9a:d0:88	Broadcast		42 ARP	Who has 10.0.32.119? Tell 10.0.33.129	
11 3.466873	10.0.33.243	40505 239.255.255.250	1900	213 SSDP	M-SEARCH * HTTP/1.1	Chat
12 3.519921	10.0.32.219	60845 239.255.255.250	1900	216 SSDP	M-SEARCH * HTTP/1.1	Chat
13 4.363370	HuaweiTe 56:73:96	Spanning-tree-(for		119 STP	MST. Root = 32768/0/30:d1:7e:b2:da:77 Cost = 10000 Port = 0x8019	
14 4,388096	AsustekC 9a:d0:88	Broadcast		42 ARP	Who has 10.0.32.119? Tell 10.0.33.129	
Address Resolution	on Protocol (request)					
0000 <i>46 66 66 66</i>	66 66 44 0- Ek ek er d					
0000 11 11 11 11 1	11 11 44 8a 5b 6b c5 d 88 81 44 8a 5b 6b c5 d	e 08 06 00 01D. [kS.				
0020 00 00 00 00 00	00 00 0a 00 20 77 00 0	0 00 00 00 00 w				
0030 00 00 00 00 0	00 00 00 00 00 00 00 00					
Proha 02092016					Parkets: 200505 (Displayed: 200505 (100,0%) + Load Hime: 0	12 610
- FIODA 08092010	U				Il Fackets' snggap, Dishiaked; snggap (100'0.46), road nije; o	1.2.019

Slika 6.1.7.Izgled glavnog prozora uz prikaz snimljenih paketa

Tri podprozora (lista paketa, detalji o samom paketu i biti paketa) su međusobno povezana. Ako želimo da vidimo detalje i sadržaj paketa, potrebno je da kliknemo (obeležimo) paket u listi paketa. Klikom na detalje paketa, Wireshark automatski obeležava i bite koji odgovaraju tom delu paketa.

Prvi (gornji) podprozor glavnog prozora prikazuje spisak svih paketa koji su snimljeni. Standardne kolone prikazane u listi paketa su:

- No.-broj snimljenog paketa
- Time –relativno vreme kada je paket snimljen
- Source–izvor snimljenog paketa
- **Destination**-odredište snimljenog paketa
- Protocol-protokol paketa
- Info-informacije koje se nalaze u samom paketu

Po potrebi, možemo dodati i nove kolone koje bi nam mogle biti od interesa za analizu. Klikom na Edit->Preferences..., otvara se prozor prikazan na slici 6.1.8.



Slika 6.1.8. Dodavanje kolona u Preference kartici

Na kartici Preference, kada kliknemo na Columns, sa desne strane se prikazuju unapred definisane kolone sa imenom i tipom podataka koje prikazuju. Ako želimo da dodamo novu kolonu, potrebno je da kliknemo na dugme (+), na kraju liste pojaviće se nova kolona. U padajućem meniju kolone Type potrebno je da izaberemo šta želimo da prikažemo u novoj koloni i da na osnovu toga upišemo i ime nove kolone. Kada završimo dodavanje kolone, dovoljno je da kliknemo dugme ok i kolona će se pojaviti u listi paketa. Za potrebe analize, dodate su kolone koje prikazuju izvorišni i odredišni port, dužinu paketa, kao i expert info koji može da nas upozori na problematične pakete.

Redosled kolona možemo da menjamo kako nam odgovora. Ako želimo da izbrišemo neku kolonu, potrebno je da kliknemo na dugme (–) na kartici Preference ili desnim klikom na samu kolonu da izaberemo opciju *Remove This Column* iz padajućeg menija.

Ako kliknemo na opciju Font and Colors na kartici Preference, možemo da vidimo kojom bojom se obeležava tekst, kao i polje u kojem se kuca displej filter (slika 6.1.9.). Polje u kojem kucamo displej filtar će biti obojeno zelenom bojom ako je filtar pravilno ukucan, definisan ili ako uopšte postoji. Dok kucamo tekst samog filtra, polje će biti obojeno u crveno sve dok ne unesemo pun naziv filtra. Ako završimo kucanje filtra koji želimo da primenimo na listu snimljenih paketa, a polje je i dalje crveno, to znači da smo pogrešno uneli filtar ili da takav filtar nije definisan.



Slika 6.1.9.Legenda fonta i boje koji se koriste za odgovarajući tekst i polja

Name Resolution opcija nam omogućava izbor kako želimo da nam se prikazuju odgovarajuće adrese, da li kao brojevi ili u tekstualnom obliku, ako je poznato kome pripada odgovarajuća MAC ili IP adresa.

Capture opcija nam pruža mogućnost izbora kako želimo da se prikazuju snimljeni paketi i nudi iste opcije kao i Capture->Options opcija u meniju.

Preference kartica nam pruža mogućnost da vidimo i spisak protokola koje Wireshark podržava. Dovoljno je da kliknemo na strelicu pored polja Protocols i otvoriće se lista prikazana na slici 6.1.10. Klikom na protokol možemo videti kako je svaki od njih definisan, koji port je podešen kao vid prepoznavanja itd. Po potrebi, možemo i promeniti neke od opcija za pojedine protokole, ako to želimo.

🚄 Proba 08092016.pcap			
File Edit View Go Capt	ture Analyze Statistics Te	lephony Wireless Tools Hel	p
🛋 🔳 🔬 🛞 🛄 🛅 🗙 🗎	🖸 । ९. 🗢 🗢 🕾 🕧 🛓	📃 🗏 Q, Q, Q, II	
Apply a display filter <ctrl- :<="" td=""><td>></td><td></td><td></td></ctrl->	>		
No. Time S	iource Source	e port Destination	Destination port Length Protocol Info
619 19.558018 1	.04.25.10.6	Wireshark - Preferences	7 🔽
620 19.558041 1	0.0.33.129		
621 19.990100 f	e80::10fc:f11c:2ee	Font and Colors	Domain Name Surtem
622 20.000417 M	licro-St_6b:c5:de	Capture	
623 20.120731 1	0.0.33.129	Filter Expressions	DNS TCP ports 53
624 20.165548 2	16.58.211.35	Name Resolution	DNS UDP ports 53
625 20.185054 2	16.58.211.35	 Protocols 	
626 20.195934 1	0.0.33.129	▷ 29West	Reassemble DNS messages spanning multiple TCP segments
627 20.205304 2	12.200.190.166	2dparityfec	DNS address resolution settings can be changed in the Name Resolution preferences
628 20.208330 1	0.0.33.129	6LoWPAN	
629 20.211114 1	0.0.33.129	802.11 Radiotap	
630 20.247908 1	.04.25.10.6	A-bis OML	
631 20.396483 1	.04.25.10.6	A21	
632 20.396725 1	.04.25.10.6	ACN	
Frame 5: 119 bytes on	wire (952 bits), 119 by	ACR 122	
IEEE 802.3 Ethernet		ACtrace	
Logical-Link Control		ADB	
Spanning Tree Protocol	1	ADB CS	
		ADB Service	
		ADwin	
		Aeron	
		AgentX	
		AH	
		AIM	
0000 01 80 c2 00 00 00	e0 97 96 56 73 96 00 0	ALC	
0010 03 00 00 03 02 7c	80 00 30 d1 7e b2 da	ALCAP	
0020 27 10 80 00 e0 97	96 56 73 96 80 19 01 0	AMQP	
0030 02 00 01 00 00 00	40 00 65 30 39 37 39 3		OK Cancel Help
0040 37 33 39 35 00 00	00 00 00 00 00 00 00 00 0		
00 00 00 00 00 00 00	00 00 00 00 dc 30 1/		

Slika 6.1.10.Lista protokola koje Wireshark podržava prikazana u Preference kartici

Srednji podprozor glavnog prozora sadrži hijerarhijski prikaz informacija o pojedinačnom, selektovanom paketu. Prikaz se može razgranati na više nivoa, kako bismo videli sve informacije o datom paketu (videti sliku 6.1.11.).

	Proba 08092016.pcap							
Fil	e Edit View Go	Capture Analyze Statist	ics Telephony	Wireless Tools Help				
4	I d 🙃 🗎 🗅	80 9 6 6 5	a a 🗖 É]eee m .				
	Analysis disalary files and							Everagien
	Appry a display filter <	201-7.2	1				Minute	Cxpressiun
No.	Time	Source	Source port	Destination	Destination port	Length Protocol	Info	Expert
	589 19.229800	31.24.228.243	443	3 10.0.33.129	54184	60 TCP	443 → 54184 [ACK] Seq=138 Ack=569 Win=30336 Len=0	
	590 19.238252	31.24.228.243	443	3 10.0.33.129	54183	60 TCP	443 → 54183 [ACK] Seq=138 Ack=569 Win=30336 Len=0	
	591 19.241437	Micro-St_6b:c5:de		Broadcast		60 ARP	Who has 10.0.32.119? Tell 10.0.33.36	
	592 19.242344	31.24.228.243	443	3 10.0.33.129	54185	60 TCP	443 → 54185 [ACK] Seq=138 Ack=569 Win=30336 Len=0	
	593 19.268347	216.58.211.35	443	3 10.0.33.129	58983	460 QUIC	Payload (Encrypted), Seq: 224	
	594 19.272542	HuaweiTe_56:73:96		Spanning-tree-(for		119 STP	MST. Root = 32768/0/30:d1:7e:b2:da:77 Cost = 10000 Port = 0x8019)
	595 19.281403	10.0.33.129	60028	3 212.200.190.166	53	83 DNS	Standard query 0xa485 A sharkfest.wireshark.org	
	Internet Protocol e100 = Ver 0101 = Hez D Differentiated Total Length: (Identification: P flags: 0x00 Fragment offset Time to live: 1 Protocol: UDP (Header checksum Source: 10.0.33 Destination: 21 [Source GeoIP:	Version 4, Src: 10.0. sion: 4 der Length: 20 bytes Services Field: 0x00 9 0x0f42 (3906) :: 0 28 [27] :: 0xc76 [validation + 129 2.200.190.166 Unknown]	33.129, Dst: (5) (DSCP: CS0, disabled]	212.200.190.166				
	User Datagram Prot	tocol. Sec Port: 60028	(60028) Ds	t Port: 53 (53)				
-	osci bacagram Pro		(00020); 05	c . o. c. oo (oo)				
90 90 90 90	00 90 17 ac bc 66 10 00 45 0f 42 06 20 be a6 ea 7c 06 30 00 00 00 00 00 06 40 09 77 69 72 65	cd 78 24 af 9a d0 8 0 00 80 11 6c 76 0a 0 3 5 00 31 c7 49 a4 8 0 00 97 68 61 72 6 5 73 68 61 72 6b 03	8 08 00 45 0 0 21 81 d4 c 5 01 00 00 0 b 66 65 73 7 f 72 67 00 0	0l.x\$E. 8 .E.Blv! 1 .5.1 .I 4s harkfest 0 .wiresha rk.org				

Slika 6.1.11.Hijerarhijski prikaz informacija o pojedinačnom paketu

Donji podprozor glavnog prozora prikazuje pakete kako oni zaista izgledaju dok putuju kroz mrežu, bez bilo kakvog obrađivanja.

Svaki paket je prikazan u određenoj boji, u skladu sa protokolom za koji se koristi. Obeležavanje u bojama omogućava znatno lakši pregled liste paketa koji su snimljeni, uz jasno razlikovanje protokola, bez gledanja same kolone u kojoj je ispisan protokol. Ova opcija je dosta korisna ako treba da analiziramo fajl sa velikim brojem snimljenih paketa.

Ako kliknemo na View->Coloring Rules... opciju u meniju, otvara se prozor prikazan na slici 6.1.12.

Wireshark · Coloring Rules · Defa	ult 🔋 💌
Name	Filter
Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update
✓ HSRP State Change	hsrp.state != 8 && hsrp.state != 16
Spanning Tree Topology Char	nge_stp.type == 0x80
OSPF State Change	ospf.msg != 1
ICMP errors	icmp.type eq 3 icmp.type eq 4 icmp.type eq 5 icmp.type eq 11 icmpv6.type eq 1 icmpv6.type eq 2 icmpv6.type eq 3 icr
ARP ARP	arp
ICMP	icmp icmpvб
TCP RST	tcp.flags.reset eq 1
SCTP ABORT	sctp.chunk_type eq ABORT
TTL low or unexpected	(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp
Checksum Errors	eth.fcs_bad==1 ip.checksum_bad==1 tcp.checksum_bad==1 udp.checksum_bad==1 sctp.checksum_bad==1 mstp.chec
SMB	smb nbss nbns nbipx ipxsap netbios
I HTTP	http tcp.port == 80 http2
IPX	ipx spx
☑ DCERPC	dcerpc
Routing	hsrp eigrp ospf bgp cdp vrrp carp gvrp igmp ismp
TCP SYN/FIN	tcp.flags & 0x02 tcp.flags.fin == 1
V ICP	tcp
UDP	uap
IV Broadcast	eth[0] & 1
<	III • • • • • • • • • • • • • • • • • •
Double click to edit. Drag to move. Rules a	re processed in order until a match is found.
+ - Pa Foregrou	und Background
	OK Cancel Import Export Help

Slika 6.1.12.Obeležavanje bojama odgovarajućih paketa

U ovom prozoru možemo definisati sopstvene boje za odgovarajuće filtre ili de modifikujemo postojeće obeležavanje. Klikom na polje (+) možemo dodati novu opciju za obeležavanje paketa na osnovu filtra koji nam odgovara za analizu. Klikom na polje (-) možemo izbrisati postojeće pravilo za obeležavanje paketa.

Klikom na npr. TCP RSTpri dnu kartice prikazano je trenutno podešeno obeležavanje slova i pozadine za navedeni filtar *tcp.flags.reset eq 1*.Bilo koja od ovih opcija i boja može se promeniti klikom na *Foreground* ili *Background* polje.

6.2. Rad sa snimljenim paketima u Wireshark programskom alatu

Wireshark dozvoljava rad sa snimljenim fajlovima i njihovu kasniju analizu. Fajl za analizu čak i ne mora da se snimi na uređaju na kojem će se raditi analiza. Postoji mogućnost i da se više snimljenih fajlova skupi (*merge*) u jedan veći fajl, uz korišćenje opcije File->Merge... u meniju.

Kako bismo sačuvali snimljene pakete, potrebno je selektovati opciju File->Save As u meniju. Tom prilikom možemo odabrati lokaciju (folder) na kojoj želimo da sačuvamo snimljene pakete i u kom formatu. Ako ne preciziramo u kom formatu želimo da sačuvamo fajl, Wireshark će sačuvati fajl u default .pcap formatu. Takođe, imamo opciju i da odaberemo da li ćemo sačuvati sve prikupljene pakete ili samo one koji odgovaraju filtru koji smo primenili.

Pored opcije čuvanja snimljenih paketa, možemo ih i eksportovati u nekoliko različitih formata za analizu u drugim programima. Potrebno je selektovati opciju File->Export Packet Dissections->(As Plain Text/As CSV/As "C" Arrays/As XML).

Vreme je od izuzetne važnosti, pogotovo u analizi paketa. Svaki paket koji Wireshark snimi ima svoj vremenski pečat koji je primenjen od strane operativnog sistema. Wireshark može da prikaže tačno vreme kada je paket snimljen, kao i vreme u odnosu na sam početak snimanja prvog ili poslednjeg paketa. Spisak opcija za prikazivanje vremena snimanja paketa dobijamo klikom na opciju File->Export u meniju i prikazan je na slici 6.2.1.



Slika 6.2.1.Izbor formata vremenskog prikaza

Ako želimo neki paket da koristimo kao referentni u vremenskom smislu, potrebno je da kliknemo na opciju Edit->Set/Unset Time Reference u meniju.

Podešavanje referentnog vremena paketa je korisno samo u situacijama kada je podešeno prikazivanje vremena u odnosu na sam početak snimanja paketa. Bilo koje drugo podešavanje vremenskog formata, uz odabir referentnog vremena, samo bi dovelo do rezultata koji nisu korisni za analizu, a mogu da unesu i dodatnu zabunu.

U Wireshark aplikaciji možemo koristiti i različite filtre kako bismo jasno definisali koje pakete želimo da vidimo i analiziramo. Filtri se mogu podeliti u dve glavne grupe:

Capture filtri – definišu se prilikom samog snimanja paketa i snimiće samo one pakete koji odgovaraju datom kriterijumu. Ako unapred znamo šta želimo da snimimo ili koji tip saobraćaja ili protokola nam nije od interesa za dalju analizu, možemo da smanjimo broj snimljenih paketa i tako povećamo efikasnost. Capture filtri koriste BPF (*Berkeley Packet Filter*) sintaksu. Složenije izraze možemo dobiti korišćenjem logičkih operatora: logičko i – AND (&&), logičko ili – OR (||) i logičko ne (!). Možemo filtrirati pakete na osnovu MAC adrese, IP adrese ili DNS hostname. Takođe, možemo filtrirati pakete koji dolaze sa određenog izvora (src) ili odredišta (dst). Filtriranje na osnovu porta koji se koristi može biti korisno ako želimo da filtriramo pakete na osnovu servisa ili aplikacije (npr. port 80 za snimanje HTTP paketa). Ako nismo sigurni koji port koriste protokoli, možemo da filtriramo na osnovu samog protokola – dovoljno je da unesemo ime

protokola u polje za filtriranje (npr. *icmp, tcp, udp* itd.). Pakete možemo filtrirati i na osnovu tipa polja koje se nalazi u zaglavlju paketa.

2) Display filtri – primenjuju se na već snimljene pakete kako bismo sakrili neželjene pakete ili posmatrali samo pakete koji odgovaraju datom kriterijumu. Display filtri se češće koriste zato što ne izostavljaju pakete koji ne odgovaraju datom kriterijumu u snimljenom fajlu. Na taj način, ako želimo da vidimo i druge pakete koji su snimljeni, dovoljno je da kliknemo na dugme Clear kako bismo se vratili na originalni fajl sa svim snimljenim paketima. U svakom slučaju, mnogo je bolje da filtriramo pakete privremeno, nego da ih obrišemo ili uopšte ne snimimo. Sintaksa za display filtre je potpuno ista kao i za capture filtre.

Izrazi koji se koriste za filtriranje mogu da se iskombinuju klikom na dugme Expression..., čime se otvara prozor prikazan na slici 6.2.2.

eld Name		Relation
tcp.options.snack.sequence · Expert Info tcp.options.snack.size · TCP SNACK Size tcp.options.tfo · Fast Open Cookie tcp.options.tfo.cookie · Fast Open Cookie Request tcp.options.tfo.request · Fast Open Cookie Request tcp.options.type · Type tcp.options.type · Copy on fragmentation tcp.options.type.copy · Copy on fragmentation tcp.options.type.copy · Copy on fragmentation tcp.options.user_to - TCP User Timeout tcp.options.user_to_granularity · Granularity tcp.options.user_to_roal. · User Timeout tcp.options.user_to_val · User Timeout tcp.options.wscale.shift · Shift count tcp.options.wscale.shift · Shift count tcp.pdu.last_frame · Last frame of this PDU tcp.pdu.size · PDU Size tcp.pdu.size · PDU Size tcp.pdu.time · Time until the last segment of this PDU tcp.porc.dstpid · Destination process name tcp.proc.dstpid · Destination process ID tcp.proc.dstuid · Destination process user ID tcp.proc.srccid · Source process name tcp.proc.srccid · Source process user ID tcp.proc.srccid · Source process user ID tcp.proc		is present == != > < < > > < > > So Predefined Values
	*	Range (offset:length)
earch:		
:p.port == 80		
lick OK to insert this filter		

Slika 6.2.2.Kartica za pravljenje izraza za display filtre

U prvom delu kartice možemo da izaberemo protokol i u ponuđenoj listi kriterijum na osnovu koga želimo da filtriramo pakete. Nakon toga, biramo relaciju koja nam je potrebna (jednako je ==, nije jednako !=, veće je >, manje je <, itd.). Na kraju, unosimo brojnu vrednost u vidu IP adrese, broja porta i tome slično. Kako biramo odgovarajuće delove izraza koji će predstavljati filter, pri dnu prozora se ispisuje izraz koji time dobijamo (u ovom slučaju je ilustrovano kako se dobija filtar za tcp.port==80). Filter se definiše klikom na dugme OK.

Ako pojedine filtre koristimo često, možemo i da ih sačuvamo, kako ne bismo svaki put kucali isti izraz. Odmah pored dugmeta Expression..., nalazi se dugme (+) koje nam pruža mogućnost da kreiramo *display* filtar kao dugme. Kasnije taj filtar možemo pozvati jednostavnim klikom na dugme, bez kucanja celog izraza.

Sam Wireshark dolazi sa unapred ugrađenim filtrima koji se mogu pogledati klikom na opciju Analyze->Display Filters u meniju. Dobijamo prozor prikazan na slici 6.2.3., sa listom filtara koji su već definisani.

Name	Filter	
Ethernet address 00:00:5e:00:53:00	eth.addr == 00:00:5e:00:53:00	
Ethernet type 0x0806 (ARP)	eth.type == 0x0806	
Ethernet broadcast	eth.addr == ff:ff:ff:ff:ff	
No ARP	not arp	
Pv4 only	ip	
Pv4 address 192.0.2.1	ip.addr == 192.0.2.1	
Pv4 address isn't 192.0.2.1 (don't use != for this!)	!(ip.addr == 192.0.2.1)	
Pv6 only	ίρνδ	
Pv6 address 2001:db8::1	ipv6.addr == 2001:db8::1	
PX only	ipx	
CP only	tcp	
JDP only	udp	
Von-DNS	!(udp.port == 53 tcp.port == 53)	
CP or UDP port is 80 (HTTP)	tcp.port == 80 udp.port == 80	
НТТР	http	
lo ARP and no DNS	not arp and !(udp.port == 53)	
Non-HTTP and non-SMTP to/from 192.0.2.1	ip.addr == 192.0.2.1 and not tcp.port in {80 25}	
+ - Pa		
		OK Cancel Help

Ako želimo da dodamo novi filtar na listu, možemo da kliknemo na dugme (+) i unesemo odgovarajući izraz i ime samog filtra.

6.3. Napredne opcije Wireshark programskog alata

Za mrežnu komunikaciju potreban je protok podataka između najmanje dva uređaja. Krajnja tačka (*Endpoint*) je uređaj koji šalje ili prima podatke na mreži. U zavisnosti od sloja OSI modela na kojem se odigrava razmena paketa, krajnje tačke mogu biti definisane svojom MAC ili IP adresom.

Ako kliknemo na opciju Statistics->Endpoints u meniju, biće prikazane informacije o broju paketa razmenjenih između dve kranje tačke. Wireshark nudi pet različitih tabova u kojim se može ispratiti saobraćaj između kranjih tačaka, i to na osnovu protokola: Ethernet, IPv4 i IPv6, kao i TCP i UDP.

Izlistane vrednosti možemo sačuvati u CSV ili YAML formatu klikom na dugme Copy, za dalju analizu saobraćaja. Klikom na Endpoint Types dugme možemo dodati i druge tipove mrežnog saobraćaja koji bismo želeli da analiziramo između krajnjih tačaka.

Za dodatne informacije možemo da kliknemo na opciju Statistics->Conversations u meniju. Dobićemo prikaz IP adresa krajnjih tačaka, kao i broj paketa i bita koji je poslat sa oba krajnja uređaja. Desnim klikom na bilo koje polje možemo kreirati filter baziran na tom polju (npr. filtriranje saobraćaja koji razmenjuju krajnje tačke, u zavisnosti od smera komunikacije).

Prilikom rada sa velikim fajlovima snimljenih paketa, ponekad je neophodno da se ustanovi koji procenat saobraćaja odlazi na koji protokol. Informacije o zastupljenosti pojedinačnih protokola u snimljenim paketima možemo dobiti klikom na opciju Statistics->Protocol Hierarchy. Svaki paket može sadržati više protokola koji pripadaju različitim slojevima. Ovaj prozor je najčešće prvi koji se gleda prilikom analize paketa.

Prilikom analize snimljenih paketa, možemo izabrati opciju da Wireshark prevede adrese za koje ima informacije (umesto MAC adrese prikazuje ime uređaja ili umesto IP adrese prikazuje veb adresu). Na ovaj način možemo olakšati samu analizu i lakše identifikovati saobraćaj koji potiče sa tih prevednih adresa koje su lakše za pamćenje.

Dissector protokola pruža mogućnost formatiranja paketa koji se prenose preko mreže u vidu protokola. *Dissector* možemo da gledamo kao neku vrstu prevodioca između podataka koji se šalju reko mreže i samog programskog alata Wireshark. Kako bi protokol bio podržan od strane Wireshark aplikacije, mora da ima ugrađen *dissector* u okviru same aplikacije. Moguće je napisati i odgovarajući *dissector* za neki novi ili nedefinisani protokol u odgovarajućoj skripti.

Postoji mogućnost i da se unapred ugrađeni *dissector* modifikuje kako će Wireshark da interpretira pojedine pakete. To je ponekad neophodno kod protokola koji koriste neki drugi port pored onog koji je definisan u *dissector* okviru. Wireshark u tim situacijama neće prepoznati protokol kao takav ili će prikazati kao da se radi o nekom drugom protokolu.

Dovoljno je da desnim klikom izaberemo paket čiji sadržaj nije pravilno interpretiran i kliknuti na opciju *Decode As* u padajućem meniju. Na osnovu broja porta možemo promeniti kojoj aplikaciji pripada dati paket i na taj način ga pravilno interpretiramo.

Bitno je napomenuti da Wireshark ne čuva forsirano dekodiranje nakon što se sačuva snimljeni fajl i zatvori sama aplikacija. Neophodno je da se kreira forsirani decoder svaki put kada se snimljeni fajl otvori.

Izvorni kod svakog *dissector*-a možemo naći na Internetu u folderu epan/dissectors. Svaki od njih je obeležen na sledeći način – *packets-protocolname.c.* Pregledanjem koda možemo otkriti na koji način Wireshark razlikuje protokole, kao i zbog čega dolazi do grešaka prilikom interpretacije.

Jedna od funckija koje nudi Wireshark je praćenje TCP streama u lako čitljivom formatu. Potrebno je desnim klikom da izaberemo paket čiji TCP stream želimo da pratimo i da kliknemo na opciju Follow->TCP Stream u padajućem meniju. TCP Stream se otvara u posebnom prozoru prikazanom na slici 6.3.1. Možemo da primetimo da je tekst prikazan u dve različite boje. Tekst u crvenoj boji označava saobraćaj od izvora do odredišta, dok tekst u plavoj boji predstavlja saobraćaj u obrnutom smeru, od odredišta do izvora. Bojom se obeležava koja strana je inicirala komunikaciju.

Veličina pojedinačnog paketa ili grupe paketa može nam reći dosta toga o samoj komunikaciji. U normalnim uslovima, maksimalna vrednost okvira (*frame*) u Ethernet mreži je 1518 bajtova. Kada oduzmemo Ethernet, IP i TCP zaglavlje, ostaje oko 1460 bajtova koji se mogu koristiti za prenos zaglavlja ili podataka aplikacionog sloja.

Klikom na opciju Statistics->Packet Lengths otvara se prozor koji nam prikazuje ukupan broj paketa, kao i procenat manjih i većih paketa. Manji paketi obično sadrže kontrolne sekvence protokola i najčešće ne prenose same podatke. Veći paketi obično sadrže veliki broj podataka koji se prenosi.

Vireshark · Follow TCP Stream (tcp.stream eq 13) · Prol	ba 08092016	
BFC.QS.wr.	".+./.,.0	
/.5.		ļ
	#	
h2	2.http/1.1uP	
fY '	′o0+"#h2	
00FQ.9Ku'0		
.*.H.=01.0UGB1.0U	.Greater Manchester1.0USalford1.0U.	
COMODO CA Limited1806U/COMODO ECC D	Domain Validation Secure Server CA 20	
i0327000000Z.		
51002235959Z011!0UDomain Control V	/alidated1!0UPositiveSSL Multi-	
<pre>main1\$0"Ussl279044.cloudflaressl.c</pre>	:om0Y0*.H.=*.H.=BL".R6.t.m.,h.	
	00U.#0@ agqO,o+v=.0Uif5	k^.0?
UU	+++	tps://
cure.comodo.com/CPS0g0VU00M	<pre>40K.I.G.Ehttp://crl.comodoca4.com/</pre>	
MODOECCDomainValidationSecureServerCA2.c	r10+	
MODOECCDomainValidationSecureServerCA2.c	rt0%+0http://ocsp.comodoca4.com0U	
ssl279044.cloudflaressl.com*.brand	victoria.com*.catholicmajority.com*.contractcars.com*.lov	ecamden.org*.nam
heckr.com*.newportbeachindy.com*.phi	ilthymag.com*.psychsignal.com*.squawkrbot.com*.transfermet	odvd.com.
winpcap.org*.wireshark.orgbrandvicto	oria.comcatholicmajority.comcontractcars.comlovecamden.org	namecheckr.com
wportbeachindy.comphilthymag.compsyc	chsignal.comsquawkrbot.comtransfermetodvd.comwinpcap.org.	
reshark.org0	5	
*.H.=H.0E.!n[KLeV	./#.v Rou.~J.1.F*6eE&00%.	[%.i&Uf.
T.0		L
*.H.=01.0UGB1.0U	Greater Manchester1.0USalford1.0U.	
COMODO CA Limited1+0)U"COMODO ECC C	Certification Authority0	
09250000007.		
092423595970 1.0 U GB1.0 U	Greater Manchester1 0 U. Salford1 0 U.	
COMODO CA Limited1806. U/COMODO ECC D	Omain Validation Secure Server CA 20Y0*.H.=*.H.=B	8i.nY
v * ka! >1.3 *C d %P \$ 18	3 % G H f f b l H t f ug H AG Hw v II f ag	
		$aca > - \cdot http://$
<pre>comedeca com/COMODOECCCentificationAut</pre>	therity only + fade: + a /http://ont compares.com	/
MODOECCAddToustCA coto% + 0 http:	//ocsp. comodoca4.com0	V
* H h @ 1 hc% 0 V 7 70 -	///ocsp.comodoca+.como	
		DNG
* u	.Jua.1.0	
	TTD Networks"ON UNABLE Steam 1 CA Basto	
Add	the second	
client pkt(s), 17 server pkt(s), 17 turn(s).		
dient pkt(s), 17 server pkt(s), 17 turn(s).		Channel 12

Slika 6.3.1. Prozor koji prikazuje TCP stream

Najbolji način za analizu snimljenih paketa dobijamo uz pomoć grafika. Wireshark sadrži nekoliko opcija za grafičku obradu snimljenih podataka.

IO Graphs nam pruža mogućnost grafičke obrade snimljenih paketa aplikacija ili protokola. Potrebno je obeležiti bar jedan paket pre nego što kliknemo na opciju Statistics->IO Graphs u meniju. Nakon toga otvara se prozor na kojem će biti iscrtani odgovarajući grafici. Postoji mogućnost da posmatramo više grafika istovremeno i na taj način poredimo različite protokole. Možemo da menjamo *display* filtar (uz odgovarajuću boju) za svaki grafik pojedinačno u zavisnosti od toga šta želimo da prikažemo na njemu. Pored toga, možemo birati šta će biti prikazano na X i Y osi grafika.

Round-Trip TimeGraphing nam ilustruje RTT (kao što samo ime kaže) za bilo koji snimljeni paket. RTT predstavlja vreme koje je potrebno da paket dođe do svog odredišta i da se pošalje potvrda da je paket stigao. Analiza RTT najčešće se primenjuje ako želimo da otkrijemo spore tačke ili zagušenja do kojih može doći u toku komunikacije. RTT grafik se otvara klikom na opciju Statistics->TCP Stream Graphs->Round Trip Time Graph. Svaka tačka na RTT grafiku predstavlja RTT pojedinačnog paketa. Klikom na pojedinačnu tačku na grafiku vodi nas direktno do pozicije paketa u listi paketa glavnog prozora.

Flow Graphing opcija je veoma korisna za vizuelizaciju konekcije i prikaz prenosa podataka u vremenu. Sastoji se od kolona koje predstavljaju hostove i strelica koje prikazuju u kom smeru se odvija komunikacija. Otvara se klikom na opciju Statistics->Flow Graphs.

Dissector svakog protokola u Wireshark aplikaciji definiše polje *expert info* koje se može koristiti za upozorenje određenih delova paketa koji koristi dati protokol. Informacije se dele u četiri kategorije:

- Chat osnovne informacije o komunikaciji
- Note neobični paketi koji mogu biti deo normalne komunikacije
- Warning neobični paketi koji najčešće nisu deo normalne komunikacije
- Error greška u paketu (na osnovu procene *dissector*-a)

Poželjno je uneti novu kolonu u glavnom prozoru koja će sadržati informacije ovog tipa. Pored toga, klikom na opciju Analyze->Expert Information otvara se prozor koji izlistava sve greške ili sumnjive delove paketa.

7. Rezultati analize snimljenog saobraćaja

Paketi su snimljeni na personalnom računaru u radnom okruženju (međusobna veza između korisnika, pristup zajedničkom serveru, pristup eunet mejl serveru, onlajn pristup štampaču). Nakon instalacije programskog alata Wireshark, pokrenuto je snimanje paketa na računaru koji je povezan na postojeću LAN mrežu. Prilikom snimanja paketa urađen je ping IP adrese 8.8.8.8. koja priprada Gugl kompaniji. Pristupano je raznim stranicama na Internetu, ostvareno skidanje video sadržaja sa sajta <u>https://www.youtube.com/?hl=sr&gl=RS</u>, pristupano onlajn bazama podataka austrijskih mobilnih operatera i skidan sadržaj sa njih.

Svrha ovog poglavlja je analiza snimljenih paketa, uz grafičke ilustracije, kako bismo mogli da vidimo tip paketa koji se razmenjuje, njihov sadržaj, IP adrese, portove, protokole koji se koriste, ukupan i prosečan broj različitih paketa. Analiza je rađena na osnovu *.pcap fajla koji je sačuvan kao Proba 07092016.

Klikom na opciju Statistics->Protocol Hierarchy, otvara se prozor prikazan na slici 7.1.1.

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits
4 Frame	100.0	748043	100.0	668523973	5820 k	0	0	0
4 Ethernet	100.0	748043	100.0	668523973	5820 k	0	0	0
Logical-Link Control	0.1	433	0.0	51527	448	0	0	0
Internet Protocol Version 6	0.2	1241	0.0	283113	2464	13	1066	9
Internet Protocol Version 4	99.3	742915	99.9	667989597	5815 k	0	0	0
Address Resolution Protocol	0.5	3467	0.0	200802	1748	3467	200802	1748

Slika 7.1.1. Hijerarhijski pregled protokola koji idu preko Ethernet protokola

U koloni protokol prikazani su protokoli koje je Wireshark uspeo da prepozna. U koloni paketi prikazan je ukupan broj paketa, kao i broj paketa po svakom protokolu. Na osnovu tih vrednosti, Wireshark računa procenat paketa po svakom protokolu u odnosu na ukupan broj paketa.

Ukupan broj snimljenih paketa je 748043. Svi paketi se prenose preko Ethernet protokola sloja 2. Možemo da primetimo da najveći broj paketa sloja 3 pripada mrežnom protokolu IPv4 – 99,3% (742915 paketa), a manji deo paketa pripada ARP protokolu – 0,5% (3467 paketa), IPv6 protokolu – 0,2% (1241 paketa) i LLC protokolu – 0,1% (433 paketa).

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/
▲ Frame	100.0	748043	100.0	668523973	5820 k	0	0	0
4 Ethernet	100.0	748043	100.0	668523973	5820 k	0	0	0
Logical-Link Control	0.1	433	0.0	51527	448	0	0	0
Internet Protocol Version 6	0.2	1241	0.0	283113	2464	13	1066	9
Internet Protocol Version 4	99.3	742915	99.9	667989597	5815 k	0	0	0
User Datagram Protocol	10.1	75509	9.6	64294532	559 k	0	0	0
Transmission Control Protocol	89.2	667296	90.3	603688229	5256 k	563634	461498503	4018 k
Internet Group Management Protocol	0.0	94	0.0	5652	49	94	5652	49
Internet Control Message Protocol	0.0	16	0.0	1184	10	16	1184	10
Address Resolution Protocol	0.5	3467	0.0	200802	1748	3467	200802	1748

Slika 7.1.2. Hijerarhijski pregled protokola koji idu preko IPv4 protokola

Ako kliknemo na trougao pored IPv4 protokola, otvara se padajući meni koji nam pokazuje koji procenat paketa pripada pojedinačnim transportnim protokolima (slika 7.1.2.). TCP je u ovom

snimljenom fajlu dominantan transportni protokol sa 89,2% (667296 paketa), dok je broj UDP paketa znatno manji – 10,1% (75509 paketa). Pored TCP i UDP protokola, u padajućem meniju prikazani su i ICMP i IGMP protokoli koji imaju zanemarljiv udeo u saobraćaju (ICMP 16 paketa i IGMP 94 paketa).

Ako kliknemo na trougao pored UDP protokola, otvara se padajući meni koji nam pokazuje koji procenat paketa pripada pojedinačnim protokolima viših slojeva (slika 7.1.3.). Najveći broj paketa pripada QUIC (*Quick UDP Internet Connections*) sa 9,5% (70755 paketa), slede HTTP sa 0,2% (1587 paketa), NetBIOS Name Service sa 0,1% (709 paketa), DNS sa 0,1% (688 paketa), Link-local Multikast Name Resolution sa 0,1% (553 paketa), dok ostali protokoli imaju manje od 0,1% udela u UDP paketima.

Wireshark · Protocol Hierarchy Statistics · Proba 07092016

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
4 Frame	100.0	748043	100.0	668523973	5820 k	0	0	0
4 Ethernet	100.0	748043	100.0	668523973	5820 k	0	0	0
Internet Protocol Version 4	99.3	742915	99.9	667989597	5815 k	0	0	0
Transmission Control Protocol	89.2	667296	90.3	603688229	5256 k	563634	461498503	4018 k
User Datagram Protocol	10.1	75509	9.6	64294532	559 k	0	0	0
QUIC (Quick UDP Internet Connections)	9.5	70755	9.5	63343858	551 k	70755	63343858	551 k
Hypertext Transfer Protocol	0.2	1587	0.1	362494	3156	1587	362494	3156
NetBIOS Name Service	0.1	709	0.0	66236	576	709	66236	576
Domain Name System	0.1	688	0.0	135693	1181	688	135693	1181
Link-local Multicast Name Resolution	0.1	553	0.0	37529	326	553	37529	326
Dropbox LAN sync Discovery Protocol	0.1	516	0.0	107044	931	516	107044	931
Data	0.0	229	0.0	107489	935	229	107489	935
Multicast Domain Name System	0.0	221	0.0	79373	691	221	79373	691
NetBIOS Datagram Service	0.0	127	0.0	29909	260	0	0	0
Simple Network Management Protocol	0.0	63	0.0	7560	65	63	7560	65
Bootstrap Protocol	0.0	47	0.0	16195	141	47	16195	141
Teredo IPv6 over UDP tunneling	0.0	13	0.0	1066	9	0	0	0
Service Location Protocol	0.0	1	0.0	86	0	1	86	0
Internet Group Management Protocol	0.0	94	0.0	5652	49	94	5652	49
Internet Control Message Protocol	0.0	16	0.0	1184	10	16	1184	10
Address Resolution Protocol	0.5	3467	0.0	200802	1748	3467	200802	1748
Internet Protocol Version 6	0.2	1241	0.0	283113	2464	13	1066	9
D Logical-Link Control	0.1	433	0.0	51527	448	0	0	0

Slika 7.1.3.Hijerarhijski pregled protokola koji idu preko UDP protokola

Ako kliknemo na trougao pored TCP protokola, otvara se padajući meni koji nam pokazuje koji procenat paketa pripada pojedinačnim protokolima viših slojeva (slika 7.1.4.). Najveći broj paketa pripada SSL (*Secure Sockets Layer*) ili HTTPS sa 13,9% (104217 paketa), slede nedefinisani paketi sa 0,3% (2429 paketa) i HTTP sa 0,2% (1211 paketa), dok ostali protokoli imaju manje od 0,1% udela u TCP paketima.

rotocol	Perce	ent Packets	Packets	Percent	Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame		100.0	748043		100.0	668523973	5820 k	0	0	0
✓ Ethernet		100.0	748043		100.0	668523973	5820 k	0	0	0
Internet Protocol Version 4		99.3	742915		99.9	667989597	5815 k	0	0	0
Transmission Control Protocol		89.2	667296		90.3	603688229	5256 k	563634	461498503	4018 k
Secure Sockets Layer		13.9	104217		21.6	144712225	1259 k	99924	138627893	1206 k
Data		0.3	2429		0.4	2639567	22 k	2429	2639567	22 k
Hypertext Transfer Protocol		0.2	1211		0.1	837894	7295	685	436833	3803
Malformed Packet		0.0	131		0.0	128149	1115	131	128149	1115
NetBIOS Session Service		0.0	10		0.0	1647	14	0	0	0
WebSocket		0.0	3		0.0	1569	13	1	62	0
Sinec H1 Protocol		0.0	1		0.0	1434	12	0	0	0
User Datagram Protocol		10.1	75509		9.6	64294532	559 k	0	0	0
Internet Group Management Protocol		0.0	94		0.0	5652	49	94	5652	49
Internet Control Message Protocol		0.0	16		0.0	1184	10	16	1184	10
Address Resolution Protocol		0.5	3467		0.0	200802	1748	3467	200802	1748
Internet Protocol Version 6		0.2	1241		0.0	283113	2464	13	1066	9
Logical-Link Control		0.1	433		0.0	51527	448	0	0	0

Slika 7.1.4. Hijerarhijski pregled protokola koji idu preko TCP protokola

Ako unesemo u polje *display* filtra sledeći izraz *data and tcp and ip and eth and frame*, Wireshark će nam prikazati nedefinisane pakete. Možemo da primetimo da su to uglavnom TCP ili HTTP poruke koje se razmenjuju na mreži preko portova koji nisu definisani unapred.

Ako kliknemo na trougao pored IPv6 protokola, otvara se padajući meni koji nam pokazuje koji procenat paketa pripada pojedinačnim transportnim protokolima (slika 7.1.5.). UDP je u ovom snimljenom fajlu jedini transportni protokol koji se koristi za IPv6 sa 0,1% (1090 paketa). Preko UDP protokola prenose se DHCPv6 i HTTP protokoli. Pored UDP protokola, u padajućem meniju prikazan je i ICMPv6 (138 paketa).

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/
▲ Frame	100.0	748043	100.0	668523973	5820 k	0	0	0
▲ Ethernet	100.0	748043	100.0	668523973	5820 k	0	0	0
Internet Protocol Version 4	99.3	742915	99.9	667989597	5815 k	0	0	0
Address Resolution Protocol	0.5	3467	0.0	200802	1748	3467	200802	1748
Internet Protocol Version 6	0.2	1241	0.0	283113	2464	13	1066	9
User Datagram Protocol	0.1	1090	0.0	270055	2351	0	0	0
Link-local Multicast Name Resolution	0.1	412	0.0	36324	316	412	36324	316
DHCPv6	0.1	378	0.0	58281	507	378	58281	507
Hypertext Transfer Protocol	0.0	142	0.0	58797	511	142	58797	511
Data	0.0	110	0.0	81441	709	110	81441	709
Multicast Domain Name System	0.0	48	0.0	35212	306	48	35212	306
Internet Control Message Protocol v6	0.0	138	0.0	11992	104	138	11992	104
Logical-Link Control	0.1	433	0.0	51527	448	0	0	0

Slika 7.1.5.Hijerarhijski pregled protokola koji idu preko IPv6 protokola

Ako kliknemo na opciju Statistics->Endpoints, otvara se prozor prikazan na slici 7.1.6.

Ethernet: 10: Devide 202 Byde 67 TOP + 221 UDO - 1/25 Address Packets Byse A - 5 Packets Devide A - 5 Packets Devide A - 5 Difter B - A Underson, 65-02 130 0 0 5 300 0 0 5 300 Underson, 65-02 0 0 0 25 300 0 25 300 0 250 300 0 250 300 0 200 <	Wireshark • Endpoints • Proba	07092016					
Tester Int Upor 1/2	Ethomatic 101 march 102				-		
Address Parkets Bytes	Ethemet 101 1994 272	IPV6 . 61					
Websure 19532 14 1719 4 527 10 300 0 0 300 300 0 0 30	Address	Packets	Bytes	Packets A → B By	tes A → B I	Packets B → A Bytes B → A	
PAmocat,02 3 300 0 0 3 300 PAmocat,02 20 77 k 0 0 26 77 k PAmocat,01 22 77 k 0 0 26 77 k PAmocat,01 12 206 0 11 206 77 k PAmocat,01 12 206 0 12 206 77 k PAmocat,013 12 206 0 12 206 78 k PAmocat,013 12 206 0 12 206 78 k PAmocat,013 12 10 0 10 206 206 PAmocat,014 10 10 0 206 206 206 PAmocat,015 12 10 0 0 0 0 0 PAmocat,015 14 10 0	Watlow_fe:59:82	14	1719	4	527	10	1192
IP-Annola 16 64 952 0 0 84 9002 <t< td=""><td>IPv4mcast_02</td><td>5</td><td>300</td><td>0</td><td>0</td><td>5</td><td>300</td></t<>	IPv4mcast_02	5	300	0	0	5	300
PArticitatifie 25 78 0 0 226 PArticitatifie 33 37 0 0 37 8 0 0 37 8 0	IPv4mcast_16	84	5052	0	0	84	5052
IP-Anncal_ft 53 37 0 0 533 P-Anncal_ft 1 66 0 0 106 P-Anncal_ft 1 76 0 0 106 P-Anncal_ft 1 76 0 0 106 P-Anncal_ft 1 106 0 0 106 P-Anncal_ft 1 106 0 0 106 106 P-Anncal_ft 1 106 0 0 106 106 106 P-Anncal_ft 1 106 0 0 0 0 0 0 Apple_1bedde 25 240 0 0 0 0 0 0 P-Anncal_ft 118 75 118 0 0 0 0 0 0 0 P-Anncal_ft 118 75 118 0	IPv4mcast_fb	226	79 k	0	0	226	79 k
IP-Amrad, 110 13 106 0 0 13 006 0 1 006 0 1 006 0 <td>IPv4mcast_fc</td> <td>553</td> <td>37 k</td> <td>0</td> <td>0</td> <td>553</td> <td>37 k _</td>	IPv4mcast_fc	553	37 k	0	0	553	37 k _
IP-Amcad, 11/16 1.2 66 0 1 66 Spanning, 11/16 1.2 V 0 0 120 57 Spanning, 11/16 1.2 V 0 0 433 51 k 0 0 72 Microsof, 11/57.20 1.4 0.4 0.4 0<	IPv4mcast_fd	13	1066	0	0	13	1066 -
Pedmack Jtiffs 1.22 45 k 0 0 123 Spanning tee ("botinge]_0 3 5 k 0 0 43 Hont aff / 2 a 'Bat 105 10 k 105 10 k 0 0 Microad Uf / 5 (2 a) 1 6 0 0 0 0 Uiteon (5 f 5 c) a 1.44 10 k 104 k 10 k 0 0 Appl: 1b d6 d0 1 70 12 k 740 0 0 0 Podmack J0 20 2 10 k 0 0 2 0 0 0 Podmack J0 20 2 10 k 0 0 2 0 0 0 Podmack J0 2002 78 8 k 0 0 42 3 k 0 0 38 k Podmack J0 2002 78 8 k 0 0 42 36 k 36 k Podmack J0 2002 2 12 k 0 0 42 36 k 36 k Podmack J0 2002 2 12 k 0 0 42	IPv4mcast_01:3c	1	86	0	0	1	86
Spanningstee (for buidge) 00 433 51 k 0 0 433 Hortardy 2, #78k1 105 10 k 105 10 k 0 0 Microsof (1:5720) 1.4 104 k 104 k 0 0 0 Apple; Libridie 25 2400 25 2400 0 0 Apple; Libridie 25 2400 0 0 0 0 Howterif 0,650k6 70 11 k 55 11 k 0 0 0 Pofmaat 02 22 140 0 0 22 140 k 0 0 22 Pofmaat 10 252 140 k 0 0 252 140 k 0 0 252 Pofmaat 11, Libridie 43 380 0 0 42 38 k	IPv4mcast_7f:ff:fa	1.720	457 k	0	0	1720	457 k
HonHair/2 24784f 105 10 k 10 k 10 k 0 Utcon/E 6/5C1 1.444 104 k 104 k 0 0 Utcon/E 6/5C1 1.444 104 k 104 k 0 0 HonHair/2 6/5S0 70 119 k 740 119 k 0 0 HonHair/2 6/5S0 70 119 k 740 119 k 0 0 HonHair/2 6/5S0 70 11 k 55 11 k 0 0 0 HonHair/2 148049 55 11 k 0 0 2 100 0 0 Poments 0. 22 140 0 0 2.52 140 0 0 300 0 0 34 Poments 0. 23 140 k 0 0 24 300 300 38 k 0 0 38 k 300 k 300 k 300 k 38 k <td< td=""><td>Spanning-tree-(for-bridges)_00</td><td>433</td><td>51 k</td><td>0</td><td>0</td><td>433</td><td>51 k</td></td<>	Spanning-tree-(for-bridges)_00	433	51 k	0	0	433	51 k
Microsoft LifsL32 1 60 0 Apple LibsCade 25 2400 0 Apple LibsCade 25 2400 0 Apple LibsCade 25 2400 0 Howsterf DisSCade 70 119k 0 0 Howsterf DisSCade 70 119k 0 0 Howsterf DisSCade 70 119k 0 0 Pofmanat DisSCade 70 114k 0 0 0 Pofmanat DisSCade 70 14k 0 0 22 140k 0 0 Pofmanat DisSCade 73 8k 0 0 34 38k 38k Pofmanat DisSCade 73 8k 0 0 42 38k 38k Pofmanat DisSCade 73 8k 0 0 42 38k 38k Pofmanat DisSCade 516 0 0 44 48k 38k 38k Pofmanat DisSCade 516 0 0 48 38k 38k Pofmanat DisSca	HonHaiPr_2e:78:bf	105	10 k	105	10 k	0	0
LikeonEcifScia 1.444 104 1004 00 Apple LibeoSe 75 2400 0 00 HewketP Doc6Se0 700 119 7400 119 00 00 Pochnact Doc6Se0 700 118 55 11 00 00 Pochnact Doc 25 140 0 0 22 100 00 Pochnact Doc 25 140 0 0 23 100 100 Pochnact Doc 25 140 0 0 252 100 100 100 Pochnact Doc 25 140 0 0 252 30000 3000 300	Microsof_bf:5f:20	1	60	1	60	0	0
Apple 1 beside 25 2400 0 0 Howler 10 6650 740 119 400 0 0 Howler 10 6650 740 119 400 0 0 0 Howler 10 6650 740 118 55 11 k 0 0 0 Pofineat 10 252 140 0 0 22 140 00 Pofineat 16 44 380 0 0 34 000 0 34 Pofineat 16 44 384 0 0 44 388 38 38 38 Pofineat 10:003 412 26 0 24 34 38 34 38 34 38 34	LiteonTe_6f:5c:1a	1.444	104 k	1444	104 k	0	0
Hewketty 66660 70 119 70 139 70 139 0 0 Howketty 66660 70 13 55 11 k 0 0 Howketty 06660 2 140 0 0 2 140 0 Pofmont 02 2 140 0 0 22 140 k 0 140 k Pofmont 02 25 140 k 0 0 252 140 k 368 k 3080 0 0 368 k 3080 368 k	Apple_1b:e6:8e	25	2400	25	2400	0	0
Honkard P1 2486:0 55 11 k 55 11 k 0 0 2 0 0 0 Pofmont Q1 22 140 k 0 0 252 0	HewlettP_06:68:00	740	119 k	740	119 k	0	0
IpAdmand 02 2 140 0 0 2 140 0 0 2 140 0 0 252 140 ko 140 ko 3080 0 0 34 3080 0 0 34 3080 0 0 34 3080 0 0 34 3080 0 0 34 3080 0 34 3080 0 34 3080 0 38 8 0 35 ko 0 35 ko 0 35 ko 0 38 ko 35 ko 0 38 ko 35 ko 0 42 36	HonHaiPr_1d:f8:d9	55	11 k	55	11 k	0	0
IpAdmast Jc 22 140 k 0 0 252 140 k 00 140 k IpAdmast Jc 34 0 0 34 000 0 34 IpAdmast Jc 48 35 k 0 0 48 0 35 k IpAdmast Jc 74 0 0 34 0 36 k	IPv6mcast_02	2	140	0	0	2	140
Promost 16 34 3000 0 0 34 3000 0 0 34 3000 0 0 34 3000 0 0 34 3000 0 0 34 3000 0 0 34 3000 0 0 3000 3000 900 3500 900 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 3500 35000 3500 350000 350000 350000 350000 350000 350000 350000 350000 350000 350000 3500000 35000000 3500000000 35000000000000000000000000000000000000	IPv6mcast_0c	252	140 k	0	0	252	140 k
Promest fib 48 35 k 0 0 48 35 k 36 k 35 k 36 k 35 k 36 k <t< td=""><td>IPv6mcast_16</td><td>34</td><td>3080</td><td>0</td><td>0</td><td>34</td><td>3080</td></t<>	IPv6mcast_16	34	3080	0	0	34	3080
Ip-Amodd, 10:1002 378 98 0 0 378 98 0 0 378 98 0 0 378 98 0 0 378 98 8 6 96 36 36 0 42 36 8 8 8 8 97 172 0 0 2 314 97 98 98 97 98 98 97 97 9 9 9 97 <t< td=""><td>IPv6mcast_fb</td><td>48</td><td>35 k</td><td>0</td><td>0</td><td>48</td><td>35 k</td></t<>	IPv6mcast_fb	48	35 k	0	0	48	35 k
IpAdmack 101303 412 3 k 0 0 412 3 k 0 0 412 3 k 0 0 42 3 k 0 0 42 3 k 0 0 2 100	IPv6mcast_01:00:02	378	58 k	0	0	378	58 k
PyAmaat_H072c20 2 172 0 0 2 172 PyAmaat_H072c20 6 515 0 0 6 515 PyAmaat_H1220c24 8 688 0 0 4 344 PyAmaat_H12720c24 8 688 0 0 1 344 PyAmaat_H12720c24 8 688 0 0 1 344 PyAmaat_H12720c24 8 688 0 0 1 344 PyAmaat_H15767c 6 516 0 0 1 366 PyAmaat_H15767c 6 516 0 0 1 36 PyAmaat_H156466 1 86 0 0 1 34 PyAmaat_H1564767c 1 8 0 0 1 34 <td>IPv6mcast 01:00:03</td> <td>412</td> <td>36 k</td> <td>0</td> <td>0</td> <td>412</td> <td>36 k</td>	IPv6mcast 01:00:03	412	36 k	0	0	412	36 k
Pyömää (f14)4/14/8 4 0 0 6 316 Pyömää (f14)4/14/8 4 4 0 0 4 314 Pyömää (f14)4/14/8 1 8 0 0 1 368 Pyömää (f14)7/14 1 8 0 0 1 368 368 Pyömää (f16)/16/14 1 8 0 0 1 368 368 Pyömää (f16)/16/14 1 8 0 0 1 368 368 Pyömää (f16)/16/14 1 8 0 0 1 368 368 Pyömää (f16)/16/14 1 8 0 0 4 344 344 368 368 Pyömää (f16)/16/14 18 1548 0 0 18 1548 374 374 374 374 374 </td <td>IPv6mcast ff:07:2c:20</td> <td>2</td> <td>172</td> <td>0</td> <td>0</td> <td>2</td> <td>172</td>	IPv6mcast ff:07:2c:20	2	172	0	0	2	172
Pyömicst ff:14:14:16: 4 4 0 0 4 344 Pyömicst ff:17:20:24 8 683 0 0 8 688 688 Pyömicst ff:17:20:24:33 1 86 0 0 1 688 Pyömicst ff:17:20:24:44:45:0 1 86 0 0 1 686 Pyömicst ff:17:20:47:0 6 516 0 0 1 686 686 Pyömicst ff:17:20:47:0 74 0 0 9 774 68 68 686 Pyömicst ff:15:20:47:0 1 154 0 14 68 686 686 686 686 686 686 686 686 686 686 686 686 686 686 686 686 686 686 686<	IPv6mcast ff:13:2b:b6	6	516	0	0	6	516
Pxdmcat_ff2/2024 8 688 0 8 688 Pxdmcat_ff2/2024 8 688 0 0 1 668 Pxdmcat_ff2/2024 186 0 0 1 668 Pxdmcat_ff2/2024 172 0 0 2 172 Pxdmcat_ff2/2024 1 1 86 0 0 166 Pxdmcat_ff2/2024 1 6 0 0 166 516 Pxdmcat_ff2/2024 4 34 0 0 4 344 Pxdmcat_ff2/2024 18 154 0 18 1548 1548 Pxdmcat_ff2/2024 18 1548 0 18 1548 1548 Pxdmcat_ff2/2024 18 1549 1549 1549 1549 Pxdmcat_ff2/2024	IPv6mcast ff:14:af:8e	4	344	0	0	4	344
IpAmast #289933 1 86 0 0 1 86 0 1 86 172 174 172 174<	IPv6mcast ff:17:20:24	8	688	0	0	8	688
Py-6mcard_#738/def58 2 172 0 0 2 172 Py-6mcard_#738/def58 2 172 0 0 2 172 Py-6mcard_#738/def58 1 86 0 0 1 6 516 Py-6mcard_#738/def58 1 86 0 0 1 6 516 Py-6mcard_#748/def58 1 86 0 0 1 6 516 Py-6mcard_#748/def58 2 172 0 0 9 774 774 Py-6mcard_#748/def58 2 172 0 0 9 774 774 Py-6mcard_#748/def58 2 172 0 0 2 774 Py-6mcard_#748/def58 2 172 0 0 2 172 Py-6mcard_#748/def58 18 154 0 18 154 154 154 Py-6mcard_#748/def58 18 154 0 18 154 154 154 Py-6mcard_#748/def58 18 154 18 154 10	IPv6mcast ff:29:59:33	1	86	0	0	1	86
Phomcat (fride/167) 6 0 0 1 66 66 517 517 517 517 517 517 517 517 517 517 517 517 517 518	IPv6mcast ff:38:dd:68	2	172	0	0	2	172
Ip-Amod_1fr67/67/e 6 516 0 0 6 516 57 57 57 57 57 57 57 57 57 57 57 57 58 58 58 58 58 58 516 50 516 50 516 50 516 57 <td< td=""><td>IPv6mcast ff:3a:99:f9</td><td>1</td><td>86</td><td>0</td><td>0</td><td>1</td><td>86</td></td<>	IPv6mcast ff:3a:99:f9	1	86	0	0	1	86
Profincat_ffddd1066 1 65 0 0 1 86 86 IPx6mcat_ffddd1066 1 65 0 0 9 774 0 0 9 774 IPx6mcat_ffdd2167 9 774 0 0 9 774 0 0 9 774 IPx6mcat_ffdd2247 9 774 0 0 4 344 344 IPx6mcat_ffdd23355 2 172 0 0 2 312 IPx6mcat_ffdd23741 18 154 0 18 154 0 185 IPx6mcat_ffdd23742 18 158 0 18 158 185 185 IPx6mcat_ffdd23741 18 158 0 18 160 195 185 IPx6mcat_ffdd23742 18 18082016.csv/ Imformat_ffdd237 18 160 196 196 196 196 196 196 120 197 196 120 197 196 120 197 196 120 197 197 197	IPv6mcast ff:67:76:7e	6	516	0	0	6	516
IpAdmod_III 68641a7 9 774 0 0 9 774 IpAdmod_III 68641a7 4 344 0 0 4 344 0 0 134 IpAdmod_III f36641a7 18 1545 0 0 2 131 134 <	IPv6mcast ff:6d:bf:06	1	86	0	0	1	86
Ip-dim:dig:1056/ster.0 4 344 0 0 4 344 Ip-dim:dig:1056/ster.0 2 172 0 0 2 172 Ip-dim:dig:105295 2 172 0 0 2 1548 IP-dim:dig:105295 18 1548 0 0 18 1548 IP-dim:dig:105295 18 1548 0 18 1548 1548 IP-dim:dig:105295 12 172 0 18 16 100 100 IP-dim:dig:105 IP-dim:dig:105 100	IPv6mcast ff:8e:a2:a7	9	774	0	0	9	774
IP-6 mcest_ff:b23395 2 172 0 0 2 172 IP-6 mcest_ff:b23395 2 172 0 0 2 154 IP-6 mcest_ff:b23395 18 154 0 0 18 1548 1548 IP-6 mcest_ff:b23395 IIIII to douby filter Image: comparison of the compari	IPv6mcast ff:96:4b:c0	4	344	0	0	4	344
INSTRUCTOR 18 1548 0 0 18 1548 1548 INSTRUCTOR Limit to dapley filter Indicate Type Indicate Type Indicate Type Microsoft Excel - 18082016.ccv Microsoft Excel - 18082016.ccv Copy Microsoft Excel - 18082016.ccv Indicate Type Image: State Sta	Py6mcast ff:b2:63:95	2	172	ō	0	2	172
Reme resolution I unit to disploy filter Indicont Type Microsoft Excel - 18002016.csv Copy • Map Copy • Map Image: Star - A IP Image: All (P) Copy • Map	IPv6mcast ff:b5:d7:e1	18	1548	0	0	18	1548 -
Image: Name resolution Limit to dapley filter Endowrit Inse Endowrit Inse Microsoft Excel - 18082016.czv Copy • Map Close Heb Image: State of the state of							
Microsoft Excel - 10062016.cvv Map Close Heb	Name resolution	mit to displ	ay filter				Endpoint Types
[Microsoft Excel - 1802016.cv]							Copy V Map Close Help
🚱 📇 🔮 🚺 📶 🔼 🔣 🚾 🚳			Mie	crosoft Excel - 18082	016.csv		
	📀 🔚 🕑	G			W	3	SR 🔺 🔏 🔐 🎼 🐗 🀠 0.52 12.09.2016

Slika 7.1.6. Krajnje tačke u komunikaciji

U njemu možemo da vidimo broj paketa koji se razmenjuje između tačke A i tačke B, uz informacije o adresi, bilo da je u pitanju MAC adrese (Ethernet kartica) ili IP adresa (kartice IPv4 i IPv6). Pored toga, možemo da pratimo saobraćaj između krajnjih tačaka za TCP i UDP protokole.

Wireshark · E	ndpoint	s · Proba (/092016						
Ethernet · 101	IPv4	4 · 272	IPv6 · 67	TCP • 721	UDP - 1715				
ddress	Port	Packets	Bytes P	ackets A → B	Bytes $A \rightarrow B$	Packets B → A	Bytes B → A	Latitude	Je Longitude
1.23.5	80	23	2241	8	962	15	1279	-	
3.150.180	80	1.052	978 k	684	936 k	378	42 1	_	_
75,83,83	443	24	8423	14	6927	10	1496	_	_
7 238 12	443	549	521 k	379	498 k	170	22 1	_	
0.2.10	445	13	1809	5	757	8	1052	_	_
0.2.14	445	10	570	5	300	5	270	-	
0.32.13	631	1	60	1	60	0	0		_
0.32,239	53748	4	264	0	0	4	264	_	_
0 33 4	34739	9	842	0	0	9	842	_	_
0.33.7	54330	2	128	2	128	0	0	_	_
0.33.7	54331	2	128	2	128	0	0	-	_
0.33.7	54291	5	345	2	108	3	237	-	-
0.33.7	54324	21	4722	11	1570	10	3152	_	_
0.33.7	54332	14	2771	8	1524	6	1247	_	_
0.33.7	54334	39	7613	19	1689	20	5924	_	_
0 33 7	54335	68	45 k	31	7561	37	37 1	_	_
0 33 7	54336	15	4469	8	767	7	3702	_	
0 33 7	54337	15	4469	8	767	7	3702	_	
0 33 7	54338	15	4469	8	767	7	3702	_	
0 33 7	54339	30	8824	16	3079	14	5745		
0 33 7	54340	3	194	3	194	0	0		
0 33 7	54341	32	3134	16	1797	16	1337	_	_
0 33 7	54342	33	3182	17	1851	16	1331	_	_
0 33 7	54343	3	194	3	194	0	0	_	_
0 33 7	54344	39	7100	19	1677	20	5423	_	_
0 33 7	54323	3	205	1	54	2	151	_	_
0 22 7	51113	44	2708	22	1241	22	1467		
0 33 7	54345	10	2397	5	737	5	1660	_	
0.227	54346	10	1052	5	618	5	434		
0 22 7	54348	13	1607	7	854	6	753		
0.227	54340	14	4770	8	1697	5	3073		
0 33 7	54350	22	5799	10	1274	12	4505	_	
10 22 7	54351	17	5858	8	1658	12	4100		
0.0.22.7	54352	13	4233	7	1664	6	2560		
Name resoluti	an	E Lin	it to displa	ny filber	1001				Endpoint Copy - Map Close -
) 🔚		3	G			W 🛛			SR 🔺 🎪 ⊮ 😵 🚓 🕕

Slika 7.1.7.Krajnje tačke u komunikaciji TCP protokola

Klikom na opciju Statistics->Conversations, otvara se prozor prikazan na slici 7.1.8.

Wiresbark Conversations Proba 07092016

ddress A	Port A Address B	Port B	Packets	Bytes	Packets A - B	Bytes A - B	Packets B → A	Bytes B → A	Rel Start	Duration	$Bits/s A \rightarrow B$	Bits/s B → A	
0.32.239	53748 52.0.253 107	4244	4	264	0	0	4	264	636.333141000	4.947693	0		42
0.0.33.4	34739 64.233.167.188	5228	9	842	0	0	9	842	841.004876000	59,885427	0		11
0.33.7	54330 192.168.2.18	9100	2	128	2	128	0	0	0.932514000	6.000301	170		
0.33.7	54331 192.168.2.16	8080	2	128	2	128	0	0	1.031491000	6.000367	170		
0.33.7	54291 91.245.214.174	443	5	345	2	108	3	237	9.828607000	0.001802	_		-
0.0.33.7	54324 198.16.79.61	80	21	4722	11	1570	10	3152	10.541708000	75.504325	166		33
0.0.33.7	54332 79.101.14.32	80	14	2771	8	1524	6	1247	10.623456000	154.213252	79		6
0.33.7	54334 216.58.214.206	443	39	7613	19	1689	20	5924	12.010183000	240.070167	56		19
0.0.33.7	54335 31.24.228.243	443	68	45 k	31	7561	37	37 k	16.287095000	136.001892	444		220
0.33.7	54336 31.24.228.243	443	15	4469	8	767	7	3702	16.287369000	12.087836	507		245
.0.33.7	54337 31.24.228.243	443	15	4469	8	767	7	3702	16.287562000	12.087461	507		245
0.33.7	54338 31.24.228.243	443	15	4469	8	767	7	3702	16.287729000	12.087842	507		245
0.0.33.7	54339 31.24.228.243	443	30	8824	16	3079	14	5745	16.955250000	135.289655	182		33
0.0.33.7	54340 192.168.2.16	8080	3	194	3	194	0	0	19.033074000	9.001675	172		
0.33.7	54341 91.245.214.185	443	32	3134	16	1797	16	1337	21.807758000	240.024530	59		4
.0.33.7	54342 172.217.16.206	443	33	3182	17	1851	16	1331	23.869731000	240.132584	61		4
.0.33.7	54343 192.168.2.18	9100	3	194	3	194	0	0	23.929679000	9.002488	172		
.0.33.7	54344 216.58.214.227	443	39	7100	19	1677	20	5423	28.340272000	240.141416	55		18
.0.33.7	54323 104.25.10.6	443	3	205	1	54	2	151	29.387438000	0.000212			-
.0.33.7	51113 66.102.1.188	5228	44	2708	22	1241	22	1467	29.463834000	869.156316	11		1
.0.33.7	54348 52.3.212.43	80	13	1607	7	854	6	753	32.507950000	110.340143	61		5
.0.33.7	54349 65.52.144.16	80	14	4770	8	1697	6	3073	32.774642000	131.374964	103		18
.0.33.7	54350 217.23.13.174	443	22	5799	10	1274	12	4525	32.782684000	110.356644	92		32
.0.33.7	54351 130.211.115.4	443	17	5858	8	1668	9	4190	33.004675000	0.703021	18 k		47
.0.33.7	54352 213.199.133.147	80	13	4233	7	1664	6	2569	33.663309000	132,849002	100		15
.0.33.7	54353 188.125.66.104	443	52	20 k	22	2543	30	18 k	33.804251000	116.411838	1/4		126
.0.33.7	54354 66.196.65.111	443	24	6792	11	925	13	5867	34.254185000	145.845017	50		32
.0.33.7	54355 66.196.65.111	443	69	39 k	29	2687	40	36 k	34.255452000	116.300603	184		251
.0.33.7	54356 104.25.11.6	443	21	5472	11	934	10	4538	38.386821000	141./12616	52		25
.0.33.7	54357 104.25.11.6	443	163	105 k	63	5692	100	99 k	38.388521000	183.076004	248		435
.0.33.7	54358 64.233.167.95	443	38	/160	18	1631	20	5529	39.481526000	240.155483	54		18
.0.33.7	54359 216.58.214.46	443	54	5331	26	2863	28	2468	39.646774000	458.968185	49		4
.0.33.7	54360 216.58.214.200	443	39	7345	19	1701	20	5644	39.694547000	240.042330	56		18
.0.33.7	54.301 192.168.2.16	8080	3	194	3	194	0	0	40.035049000	8,999854	1/2		
Name reso	ution 📃 Limit to	display fil	ter										Conversation Ty
												Conv Follow Stream Graph	Close He
												Color St Color.	

Slika 7.1.8. Prikaz komunikacije između dva uređaja kod TCP protokola

U njemu možemo da vidimo broj paketa koji se razmenjuje između dva uređaja, uz informacije o adresama oba uređaja, bilo da je u pitanju MAC adrese (Ethernet kartica) ili IP adresa (kartice IPv4 i IPv6). Ako kliknemo na karticu TCP možemo da vidimo IP adrese uređaja koji komuniciraju, portovi koji se koriste za komunikaciju, broj paketa koji je razmenjen u oba smera i u svakom pojedinačno, trajanje same komunikacije i broj bajtova koji je razmenjen.

lopic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Packet Lengths	748043	893.70	42	1514	0.8141	100%	7.0500	643.605
0-19	0	-	-	-	0.0000	0.00%	75	-
20-39	0	-	-	H	0.0000	0.00%	-	
40-79	251996	59.97	42	79	0.2742	33.69%	3.3500	473.741
80-159	43660	87.34	80	159	0.0475	5.84%	3.2800	429.935
160-319	4262	220.84	160	319	0.0046	0.57%	0.2800	372.989
320-639	3176	488.21	320	639	0.0035	0.42%	0.4700	352.421
640-1279	3495	902.16	640	1279	0.0038	0.47%	0.3200	608.292
1280-2559	441454	1458.71	1280	1514	0.4804	59.01%	3.6800	846.965
2560-5119	0	-	-	-	0.0000	0.00%	75	-
5120 and greater	0	-	ie.	iπ.	0.0000	0.00%	-	

Ako kliknemo na opciju Statistics->Packet Lengths, otvara se prozor prikazan na slici 7.1.9.

Slika 7.1.9. Veličina snimljenih paketa

Ako sačuvamo sadržaj prozora kao *.csv fajl, otvorimo ga u programu Excel, grafički možemo ilustrovati broj paketa različitih dužina (slika 7.1.10.).



Slika 7.1.10.Broj paketa različitih dužina

Na osnovu grafika i tabele, možemo zaključiti da se najveći broj paketa nalazi u opsegu 40-79 i 1280-2559 bajtova.

Klikom na opciju Statistics->Flow Graph, otvara se prozor prikazan na slici 7.1.11. U njemu možemo da pratimo razmenu paketa, smer komunikacije između odgovarajućih IP adresa, kao i tip poruke koja se šalje. Postoji mogućnost da izaberemo TCP flow, kako bismo videli razmenu poruka TCP protokola.

Wireshark	· Flow ·	Proba 07092016												
	10.0.3	3.7 19	2, 168, 2, 16	19	8.16.79.61	216.58.214.205		10.0.33.124	91.24	5.214.185	216.58.214	.227	66, 102, 1, 18	R A
	101010	192.168.2.18	91.2	245.214.174		79.101.14.32	10.0.32.13		31.24.228.243	172.217	16.206	104.25.10	.6	
0.932514	54330 -	5YN 9100												Seq = 0
1.031491	54331 -	SYN	8090											Seq = 0
6.932815	54330 -	SYN 9100		1					1				1	Seq = 0
7.031858	54331	SYN	8080		1							1		Seq = 0
9.828607	54291 🛥	RSH, ACK - L	en: 63	443	1							1		Seq = 1 Ack = 1
9.828771	54291 🛥	FIN, AC	×	443							1	1	1	Seq = 64 Ack = 1
9.828844	54291 -	ACK		• 443	1									Seq = 1 Ack = 65
9.829411	54291 -	FIN, AC	к	• 443										Seq = 1 Ack = 65
9.830409	54291 🖛	ACK	1.00	443					-		1		1	Seq = 65 Ack = 2
10.541708	54324	1 PSH	ACK-ben: 122		B 0				8					Seq = 1 Ack = 1
10.572671	54324 🛥	PSH.	ACRI-Len: 325		80									Seq = 1 Ack = 123
10.589388	54324	PSH	ACK-ben: 122		80				1		l	1		Seq = 123 Ack = 326
10.620344	54324	Port,	CVN CVN		80				1					Seq = 326 Ack = 245
10.623456	54332 -		SYN ACH			₩ 80								Seq = 0
10.624119	54332 +		ACK	<u> </u>		80					l.			Seq = 0 Ack = 1
10.624154	54332		PSH ACK - La	759		► 80								Seq = 1 Ack = 1
10.624225	54332		ACK			► 50								Seq = 1 AOK = 1
10.624818	54332		PSH, ACK - Let	n: 226		80							1	Seq = 1 Ack = 260
10.625098	54332 4	PSH,	ACK - Len: 122			50								Seq = 1 Ack = 260
10.636354	54524	PSH.	ACK - Len: 325	1	- 00									Seq = 245 Not = 651
10.000901	54324	PSH,	ACK - Len: 122	1	80		1		l.		Į.		1	Seq = 051 ACK = 307
10.776308	54324 F	PSH.	ACK - Len: 325											Seg = 976 Ark = 489
10.727679	54332		PSH, ACK - Let	ni 277		80			1	1	Î	1	1	Seq = 260 Ack = 227
10.728806	54332 a		PSH, ACK - Ler	ni 227		- 00					-	1	1	Sec = 227 Ark = 537
10.762656	54324	PSH,	ACK - Len: 122		80									Seg = 489 Ack = 1301
		i.		į.		1	1		į.	1 1	i i	i.	1	
+														*
Packet 150: Se	a = 1301 A	lck = 611												
Show: Al na	ckets	-					Flow type:	TCP Flows						Addresses: Any
Culotti Lin po	crite up						and other (
														Reset
													iave As	Close Help
((ک) 🕑		x (V		33						SR	- 🔺 🛱 🍖	1:48 ()) 12.09.2016

Slika 7.1.11.TCP Graph Flow

7.1. Analiza paketa ARP protokola

Ako u polju za display filtar ukucamo arp, Wireshark će nam prikazati samo pakete koji sadrže ARP protokol (slika 7.1.12.).

arp								Expresso
Time	Source	Source port I	Destination	Destination port	Length	Protocol	Info	Expert
748039 918.75	0152 AsustekC_9a:		Broadcast			42 ARP	Who has 10.0.32.119? Tell 10.0.33.7	
748031 918.59	2549 LiteonTe_6f:.		Broadcast			60 ARP	Who has 10.0.32.177? Tell 10.0.33.149	
748016 918.10	2476 CompalIn_8e:.		Broadcast			60 ARP	Who has 10.0.32.15? Tell 10.0.33.172	
748009 917.82	5099 AsustekC_9a:		Broadcast			42 ARP	Who has 10.0.32.119? Tell 10.0.33.7	
748001 917.59	5465 LiteonTe_6f:		Broadcast			60 ARP	Who has 10.0.32.177? Tell 10.0.33.149	
747700 916.75	0001 AsustekC_9a:		Broadcast			42 ARP	Who has 10.0.32.119? Tell 10.0.33.7	
747699 916.68	3772 LiteonTe_6f:		Broadcast			60 ARP	Who has 10.0.32.177? Tell 10.0.33.149	
747674 915.74	9984 AsustekC_9a:.		Broadcast			42 ARP	Who has 10.0.32.119? Tell 10.0.33.7	
747662 915.09	<pre>1677 LiteonTe_6f:.</pre>		Broadcast			60 ARP	Who has 10.0.32.52? Tell 10.0.33.149	
747659 915.01	2784 HuaweiTe_bc:.		Broadcast			60 ARP	Who has 10.0.33.64? Tell 10.0.32.1	
747658 915.01	2784 HuaweiTe_bc:.		Broadcast			60 ARP	Who has 10.0.32.255? Tell 10.0.32.1	
747654 914.82	5548 AsustekC_9a:		Broadcast			42 ARP	Who has 10.0.32.119? Tell 10.0.33.7	
747645 914.52	1434 HewlettP_fe:.		Broadcast			60 ARP	Who has 10.0.32.1? Tell 10.0.32.187	
747640 914.33	0509 Micro-St_6b:		Broadcast			60 ARP	Who has 10.0.32.119? Tell 10.0.32.154	
747635 914.09	1925 LiteonTe_6f:		Broadcast			60 ARP	Who has 10.0.32.52? Tell 10.0.33.149	
747617 913.56	2120 LiteonTe_6f:		Broadcast			60 ARP	Who has 10.0.32.52? Tell 10.0.33.149	
747610 913.33	0241 Micro-St_6b:		Broadcast			60 ARP	Who has 10.0.32.119? Tell 10.0.32.154	
747593 912.48	6744 Micro-St_6b:		Broadcast			60 ARP	Who has 10.0.32.119? Tell 10.0.32.154	
746939 912.08	7476 HewlettP_06:		Broadcast			60 ARP	Who has 10.0.32.1? Tell 10.0.32.49	

Opcode: request (1) Sender MC address: LiteonTe_6f:5c:1a (18:cf:5e:6f:5c:1a) Sender IP address: 10:0.33.149 Target IPAC address: 00:00:00:00:00:00:00:00:00:00:00:00 Target IP address: 10:0.32.177

Slika 7.1.12.Paketi koji sadrže ARP protokol

Klikom na paket, možemo videti detaljan sadržaj ARP paketa u srednjem podprozoru glavnog prozora. U selektovanom paketu koristi se Ethernet protokol sloja 2 i IPv4 sloja 3 OSI modela. Veličina hardverske adrese je 6 bajtova, dok je veličina mrežne IP adrese 4 bajta. Polje Opcode nam govori da se radi o ARP zahtevu. U ARP paketu šalju se MAC i IP adresa pošiljaoca i ciljana IP adresa. Polje ciljane MAC adrese je ispunjeno nulama (adresa nije poznata).

Desnim klikom na bilo koji ARP paket možemo kreirati filter (opcija Applv as Filter). Ako želimo da vidimo samo pakete koji predstavljaju ARP zahtev, klikom na liniju Opcode: request (1)->Apply as Filter prikazuju nam se rezultati filtriranja na osnovu izraza *arp.opcode==1*.

Ako želimo da nađemo paket koji predstavlja odgovor na ovaj zahtev, potrebno je da unesemo sledeći izraz u polje za display filtar – arp.opcode==2 (čime se definiše da se radi o ARP odgovoru) i IP adresu ciljanog uređaja koja u ARP odgovoru predstavlja IP adresu pošiljaoca -(arp.opcode == 2) && (arp.src.proto ipv4 == 10.0.32.1).

Ako kliknemo na opciju Statistics->IO Graph, otvara se grafički interfejs Wireshark aplikacije u kojoj možemo da prikažemo različite grafike, uz odgovarajuće filtre.

Na slici 7.1.13. prikazan je broj ARP paketa u jedinici vremena (crvenom bojom) u odnosu na ukupan broj paketa (crnom bojom) u snimljenom fajlu.



Wireshark IO Graphs: Proba 07092016

boja)

Na slici 7.1.14. prikazan je ukupan broj ARP paketa u jedinici vremena (filtar *arp*), broj ARP zahteva (*filtar arp.opcode==1*) i broj ARP odgovora (*filtar arp.opcode==2*). U snimljenom fajlu, imamo znatno više ARP zahteva nego odgovora.



Slika 7.1.14.Broj paketa koji sadrže ARP zahteve (crvena boja) u odnosu na ARP odgovore

Kako prelazimo mišem preko pikova na grafiku, u glavnom prozoru se prikazuje paket koji odgovara toj poziciji i na taj način možemo da analiziramo paket od interesa.

Ako nam ARP paketi nisu od interesa za analizu, u polju *display* filtra možemo da unesemo sledeći izraz *!arp* i ARP paketi neće biti prikazani u glavnom prozoru.

7.2. Analiza paketa IP protokola

U polje *display* filtra možemo da unesemo razne kombinacije izraza koji filtriraju i prikazuju pakete na osnovu IP adrese.

Ako unesemo npr. ip.addr = 212.200.190.166, Wireshark će nam prikazati sve pakete koji imaju datu IP adresu ili kao adresu pošiljaoca ili kao adresu primaoca (vidi sliku 7.2.1.).

Proba 07092016.pcap			
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools	Help		
📶 🔳 🧷 🕲 📙 🛗 🕱 😋 🍳 👄 🕾 🕾 🛄 🗐 🍳 Q, Q, 🖽			
ip.addr==212.200.190.166			Expression + TCP
No. Time Source Source port Destination	Destination port	Length Protocol	Info
113557 488.500642 212.200.190.166 53 10.0.33.7	61920	294 DNS	Standard query response 0x5b86 A image6.pubmatic.com CNAME pugm22000c.pubmatic.com CNAME pugm
113558 488.501602 10.0.33.7 51061 212.200.190.166	53	77 DNS	Standard query 0x4dd4 A pm-m.d.chango.com
113559 488.503637 212.200.190.166 53 10.0.33.7	51061	546 DNS	Standard query response 0x4dd4 A pm-m.d.chango.com CNAME d.chango.com.akadns.net A 208.43.247
113560 488.519147 10.0.33.7 51207 212.200.190.166	53	80 DNS	Standard query 0x50cc A showads.pubmatic.com
113564 488.521644 212.200.190.166 53 10.0.33.7	51207	301 DNS	Standard query response 0x50cc A showads.pubmatic.com CNAME showads22000c.pubmatic.com CNAME
118669 490.665351 10.0.33.7 64539 212.200.190.166	53	74 DNS	Standard query 0xd609 A www.google.com
118670 490.667742 212.200.190.166 53 10.0.33.7	64539	226 DNS	Standard query response 0xd609 A www.google.com A 216.58.211.36 NS ns1.google.com NS ns3.goog
118671 490.676354 10.0.33.7 51886 212.200.190.166	53	73 DNS	Standard query 0xf8ab A www.google.rs
118672 490.677525 212.200.190.166 53 10.0.33.7	51886	235 DNS	Standard query response 0xf8ab A www.google.rs A 216.58.211.3 NS ns3.google.com NS ns2.google
123665 497.591367 10.0.33.7 61643 212.200.190.166	53	75 DNS	Standard query 0xb0a2 A osiris.drei.com
123666 497.646066 212.200.190.166 53 10.0.33.7	61643	222 DN5	Standard query response 0xb0a2 A osiris.drei.com A 213.94.102.242 NS ns.domainnetwork.se NS n
124319 519.499168 10.0.33.7 49631 212.200.190.166	53	79 DNS	Standard query 0xa014 A clients1.google.com
124320 519.500264 212.200.190.166 53 10.0.33.7	49631	255 DNS	Standard query response 0xa014 A clients1.google.com CNAME clients.l.google.com A 216.58.214.
125388 542.149036 10.0.33.7 50758 212.200.190.166	53	71 DN5	Standard query 0x831c A i.ytimg.com
125389 542.150107 212.200.190.166 53 10.0.33.7	50758	252 DNS	Standard query response 0x831c A i.ytimg.com CNAME ytimg.l.google.com A 216.58.209.174 NS ns1
125774 560.174012 10.0.33.7 52902 212.200.190.166	53	82 DN5	Standard query 0xec65 A content.googleapis.com
125775 560.175121 212.200.190.166 53 10.0.33.7	52902	300 DNS	Standard query response 0xec65 A content.googleapis.com CNAME googleapis.l.google.com A 172.2
125798 562.188693 10.0.33.7 58287 212.200.190.166	53	78 DN5	Standard query 0xaa89 A ad.doubleclick.net
125799 562.189810 212.200.190.166 53 10.0.33.7	58287	261 DNS	Standard query response 0xaa89 A ad.doubleclick.net CNAME dart.l.doubleclick.net A 216.58.214
125800 562.199145 10.0.33.7 61644 212.200.190.166	53	72 DNS	Standard query 0xa5b8 A i9.ytimg.com
125801 562.199418 10.0.33.7 61960 212.200.190.166	53	98 DNS	Standard query 0x2a08 A r4sn-uxax3vhgn-cxbs.googlevideo.com
<u> ۲</u>	III		•
> Frame 125800: 72 bytes on wire (576 bits), 72 bytes captured (576 b	bits)		A
Ethernet II, Src: AsustekC_9a:d0:88 (78:24:af:9a:d0:88), Dst: Huawa	eiTe_bc:6c:cd	(90:17:ac:bc:6	c:cd)
Internet Protocol Version 4, Src: 10.0.33.7, Dst: 212.200.190.166			
4 User Datagram Protocol, Src Port: 61644 (61644), Dst Port: 53 (53)			
Source Port: 61644			
Destination Port: 53			
Length: 38			
Checksum: 0x53ab [validation disabled]			
[Stream index: 1182]			
Domain Name System (query)			
[Response In: 125804]			
Transaction ID: 0xa5b8			т
🔵 🎽 Proba 07092016			Packets: 748043 * Displayed: 688 (0.1%) * Load time: 0:15.791 Profile: Default
			58 . 0 (6) (6) 19:17

Slika 7.2.1.Paketi koji za adresu pošiljaoca ili primaoca imaju IP adresu 212.200.190.166

Ako želimo da pratimo samo pakete koji nam dolaze sa pomenute IP adrese, dovoljno je da unesemo sledeći izraz u polje *display* filtra – ip.src==212.200.190.166. Za pakete koji dolaze u suprotnom smeru, dovoljno je da unesemo ip.dst==212.200.190.166 u polje *display* filtra.

Filtar na osnovu IP adrese možemo podesiti i u meniju koji se pojavljuje nakon desnog klika na paket izborom opcije *Apply as Filtar*. Mogućnosti su zaista velike.

Ako želimo da vidimo pakete koji su razmenjeni između dve IP adrese, dovoljno je da unesemo izraz (*ip.src* == 10.0.33.7) && (*ip.dst* == 94.127.4.206) u polje *display* filtra. Rezultati filtriranja mogu se videti na slici 7.2.2.

4	Proba 07092016.pcap						
File	Edit View Go Capture	Analyze Sta	atistics Telephony Wireless Tools	Help			
1	🔳 🔬 🛞 斗 🕒 🖄 🖻	۹ 👄 🔿 🖻	≝ Ŧ <u>↓</u> 및 Q Q Q II				
	ip.src == 10.0.33.7) && (ip.dst =	= 94.127.4.206)					Expression + TOP
No.	Time	Source	Source port Destination	Destination port Length	Protocol	Info	Expert
-	1986 50.646425	10.0.33.7	54363 94.127.4.206	995	66 TCP	54363 + 995 -	. Chat
	1988 50.648888	10.0.33.7	54363 94.127.4.206	995	54 TCP	54363 → 995	
	1989 50.649122	10.0.33.7	54363 94.127.4.206	995	216 TLSv1	Client Hello	
	1992 50.661246	10.0.33.7	54363 94.127.4.206	995	380 TLSv1	Client Key E.	
	1997 50.875944	10.0.33.7	54363 94.127.4.206	995	54 TCP	54363 → 995	
	1999 50.878056	10.0.33.7	54363 94.127.4.206	995	91 TLSv1	Application	
	2001 50.880440	10.0.33.7	54363 94.127.4.206	995	123 TLSv1	Application	
	2003 50.882362	10.0.33.7	54363 94.127.4.206	995	107 TLSv1	Application	
	2005 50.892697	10.0.33.7	54363 94.127.4.206	995	91 TLSv1	Application	
	2008 50.894711	10.0.33.7	54363 94.127.4.206	995	91 TLSv1	Application	
	2010 50.897548	10.0.33.7	54363 94.127.4.206	995	91 TLSv1	Application	
	2012 50.900240	10.0.33.7	54363 94.127.4.206	995	91 TLSv1	Application	
	2015 50.903969	10.0.33.7	54363 94.127.4.206	995	54 TCP	54363 → 995	
	2016 50.904229	10.0.33.7	54363 94.127.4.206	995	54 TCP	54363 → 995	. Chat
	41205 427.132374	10.0.33.7	54654 94.127.4.206	995	66 TCP	54654 → 995	. Chat
	41207 427.134869	10.0.33.7	54654 94.127.4.206	995	54 TCP	54654 → 995	
	41208 427.135136	10.0.33.7	54654 94.127.4.206	995	216 TLSv1	Client Hello	
	41211 427.144278	10.0.33.7	54654 94.127.4.206	995	380 TLSv1	Client Key E.	
	41216 427.347636	10.0.33.7	54654 94.127.4.206	995	54 TCP	54654 → 995 -	
	41218 427.349712	10.0.33.7	54654 94.127.4.206	995	91 TLSv1	Application	
	41220 427.351479	10.0.33.7	54654 94.127.4.206	995	123 TLSv1	Application _	
	41222 427.353190	10.0.33.7	54654 94.127.4.206	995	107 TLSv1	Application	
	41224 427.362540	10.0.33.7	54654 94.127.4.206	995	91 TLSv1	Application	
	41226 427.364707	10.0.33.7	54654 94.127.4.206	995	91 TLSv1	Application	
	41228 427.366941	10.0.33.7	54654 94.127.4.206	995	91 TLSv1	Application	
	41230 427.369416	10.0.33.7	54654 94.127.4.206	995	91 TLSv1	Application	
	41233 427.372173	10.0.33.7	54654 94.127.4.206	995	54 TCP	54654 ÷ 995	
	41239 427.372500	10.0.33.7	54654 94.127.4.206	995	54 TCP	54654 → 995	
	41240 427.372525	10.0.33.7	54654 94.127.4.206	995	54 TCP	54654 ÷ 995	
	Frame 1986: 66 bytes on Ethernet II, Src: Asuste Internet Protocol Versio Fransmission Control Pro	wire (528 bit kC_9a:d0:88 n 4, Src: 10 tocol, Src Po	ts), 66 bytes captured (528 bit: (78:24:af:9a:d0:88), Dst: Huawe: .0.33.7, Dst: 94.127.4.206 ort: 54363 (54363), Dst Port: 9	5) LTe_bc:6c:cd (90:17:ac: 95 (995), Seq: 0, Len: 0	oc:6c:cd)		
0	Proba 07092016						Packets: 748043 * Displayed: 7602 (1.0%) * Load time: 0:14.729 Profile: Defau
6) 👸 🕘 [[]					SR 🔺 🎪 📴 🐻 ant 🕪 19:33 13:09:2016

Slika 7.2.2.Paketi koji su razmenjeni između dve IP adrese10.0.33.7 i 94.127.4.206

Ako želimo da uporedimo broj paketa u jedinici vremena koja je poslata sa IP adrese 10.0.33.7 (IP adresa računara na kojem je snimljen .pcap fajl) u odnosu na sve pakete koji su poslati u istom tom trenutku, dovoljno je da u IO graph polju za filtriranje drugog grafika unesemo izraz ip.src ==10.0.33.7 i da prikažemo zajedno sa *default* filtrom. Dobićemo grafik sa dve linije prikazan na slici 7.2.3.



vremena Svi paketi su prikazani crnom linijom, dok su paketi koji su poslati sa IP adrese 10.0.33.7 azani crvenom linijom. Na osnovu ovog grafika možemo da primetimo da paketi koji su poslati

prikazani crvenom linijom. Na osnovu ovog grafika možemo da primetimo da paketi koji su poslati sa uređaja na kojem je snimljen saobraćaj čine malo više od trećineukupnog broja paketa u datoj jedinici vremena (38,4%).

Ako nas interesuju paketi koji u sebi sadrže IP adrese verzije 6, dovoljno je da u polju *display* filtra ukucamo *ipv6* i u glavnom prozoru možemo videti koji sve protokoli koriste IPv6, kao i sadržaj samih paketa (slika 7.2.4.).

📕 Prob	ba 07092016.pcap								- 0	ĸ
File E	dit View Go Captur	e Analyze Statistics	Telephony Wireless Tools	Help						
A H	🔬 🛞 🔒 🛅 🔀 🖸	9 € ⊕ ≌ 7 !	L 🔲 🖲 🔍 🔍 💷							
ipv6								Expres	sion + TC	P
No.	Time	Source Sou	rce port Destination	Destination port Length	Protocol	Info	Expert			*
	16 1.216824	fe80::593a:7	546 ff02::1:2	547	155 DHCPv6	Solicit XID:				
	26 1.692895	fe80::1d9d:1	546 ff02::1:2	547	157 DHCPv6	Solicit XID:				
	28 1.818422	fe80::406e:1	546 ff02::1:2	547	156 DHCPv6	Solicit XID:				
	31 2.216266	fe80::593a:7	546 ff02::1:2	547	155 DHCPv6	Solicit XID:				
	38 2.817980	fe80::406e:1	546 ff02::1:2	547	156 DHCPv6	Solicit XID:				
E.	42 3.170010	fe80::247e:b	546 ff02::1:2	547	153 DHCPv6	Solicit XID:				
	53 4.216339	fe80::593a:7	546 ff02::1:2	547	155 DHCPv6	Solicit XID:				
	62 4.715103	fe80::1d9d:1	65242 ff02::1:3	5355	88 LLMNR	Standard que				
	64 4.715985	fe80::1d9d:1	61322 ff02::1:3	5355	88 LLMNR	Standard que				
	65 4.818240	fe80::406e:1	546 ff02::1:2	547	156 DHCPv6	Solicit XID:				
	79 6.418588	fe80::3a63:b	546 ff02::1:2	547	161 DHCPv6	Solicit XID:				
	99 8.216792	fe80::593a:7	546 ff02::1:2	547	155 DHCPv6	Solicit XID:				
	101 8.402136	fe80::352d:b	546 ff02::1:2	547	155 DHCPv6	Solicit XID:				
	106 8.818499	fe80::406e:1	546 ff02::1:2	547	156 DHCPv6	Solicit XID:				Ŧ
D Ethe D Inte User	ernet II, Src: Micro ernet Protocol Versi r Datagram Protocol, Source Port: 546	-St_fd:bc:c4 (d4:3d: on 6, Src: fe80::247 Src Port: 546 (546)	7e:†d:bc:c4), Dst: IPv6 7e:be0f:df9:3458, Dst: f), Dst Port: 547 (547)	mcast_01:00:02 (33:33:00 f02::1:2	:01:00:02)					
	Destination Port: 54	7								
L 1	Length: 99									
Þ	Checksum: 0x3852 [vai	lidation disabled]								
4 01/0	[Stream index: 7]									_
- Unici	Message type: Solici	t (1)								
1	Transaction ID: 0x24	2bd3								
Þ	Elapsed time									
▷ 0	Client Identifier									
⊳]	Identity Association	for Non-temporary A	lddress							
ÞF	Fully Qualified Doma:	in Name								
Þ \	Vendor Class									
Þ¢	Option Request									
							11		110000000000000000000000000000000000000	_
	Source Port (udp.srcport)	, 2 bytes					Packets: 748043 · Displayed:	1241 (0.2%) · Load time: 0:12.636	Profile: Defa	Jult
0								SR 🔺 🦀 📴 😽 📶 🕪	19:55 13.09.2016	

Slika 7.2.4. Paketi koji sadrže IP adrese verzije 6

Možemo da uporedimo grafički prikaz paketa koji koriste IPv4 adrese (plava linija) i paketa koji koriste IPv6 adrese (crvena linija) na slici 7.2.5. Znatno veći broj paketa koristi IPv4 adrese, što je i bilo za očekivati.



7.3. Analiza paketa TCP protokola

TCP protokol nam pruža velik broj mogućnosti za analizu paketa u Wireshark aplikaciji.

Ako želimo da vidimo samo pakete koji sadrže TCP protokol, dovoljno je da u polje *display* filtra unesemo *tcp* i dobićemo ispis na glavnom prozoru kao na slici 7.3.1.

A F	Proba 07092016.pcap								
File	Edit View Go Capture	Analyze Statis	tics Telephony Wireless Tools	Help					
1	= 🧟 🛞 📕 🔂 🔁	۹ 👄 😤	¥ 🛃 🗐 🔍 Q, Q, 🖩 📗						
, b	ф							Expres	ssion + TCP
No.	Time	Source	Source port Destination	Destination port Length	Protocol	Info	Expert		*
	12 0.932514	10.0.33.7	54330 192.168.2.18	9100	66 TCP	54330 → 9100	Chat		
	15 1.031491	10.0.33.7	54331 192.168.2.16	8080	66 TCP	54331 → 8080	Chat		
	88 6.932815	10.0.33.7	54330 192.168.2.18	9100	62. TCP	[TCP Retrans	Note		
		10.0.33.7	54331 192.168.2.16	8080	62 TCP	[TCP Retrans			
	118 9.828607	91.245.214.1.	. 443 10.0.33.7	54291	117 TLSv1.2	Application			
	119 9.828771	91.245.214.1.	. 443 10.0.33.7	54291	60 TCP	443 → 54291	Chat		
	120 9.828844	10.0.33.7	54291 91.245.214.1	443	54 TCP	54291 → 443			
	121 9.829411	10.0.33.7	54291 91.245.214.1.	443	54 TCP	54291 → 443	Chat		
	122 9.830409	91.245.214.1.	443 10.0.33.7	54291	60 TCP	443 → 54291			
	126 10.541708	10.0.33.7	54324 198.16.79.61	80	176 HTTP	GET /wpad.da.	Chat		
	127 10.572671	198.16.79.61	80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200.	Chat		
	131 10.589388	10.0.33.7	54324 198.16.79.61	80	176 HTTP	GET /wpad.da.	Chat		
	132 10.620344	198.16.79.61	80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200.	Chat		
	135 10.623456	10.0.33.7	54332 79.101.14.32	80	66 TCP	54332 → 80 [Chat		
	nternet Protocol Versio ransmission Control Pro Source Port: 54708 Destination Port: 443 [Stream index: 367] [TCP Segment Len: 0] Sequence number: 1064 Acknowledgment number: Header Length: 32 bytt > Flags: 0x4501 (ACK) Window size value: 65: [Calculated window si: [Window size scaling : 0 Checksum: 0x1etb [val: Urgent pointer: 0 0 Options: (12 bytes), 1 > [S50/ACK analysis]	n 4, Src: 10.0 tocol, Src Pori 2 (relative : 78458178 55 205 :e: 260820] factor: 4] idation disable No-Operation (1	.33.7, Dst: 213.94.102.242 : 54708 (54708), Dst Port: 4 sequence number) relative ack number) rd] NOP), No-Operation (NOP), SACI	43 (443), Seq: 10642, J	Ack: 78458178, Len	: 0			
•	Transmission Control Protoc	ol: Protocol					Packets: 748043 • Displayed: 66	i7296 (89.2%) · Load time: 0:17.1	Profile: Default
) 🚞 🕘	<u>i</u> 🔀						SR 🔺 🛕 📴 😼 ad 🕸	20:39

Slika 7.3.1. Paketi koji sadrže TCP zaglavlje

Pakete možemo da filtriramo u odnosu na port koji se koristi za prenos saobraćaja. Ako u polje *display* filtra unesemo *tcp.dstport* == 443, Wireshark će nam prikazati samo pakete čiji je odredišni port 443. Na isti način možemo da filtriramo i pakete čiji je izvorišni port 443, uz odgovarajući izraz *tcp.srcport* == 443. Ako želimo da vidimo pakete koji se razmenjuju preko porta 443 sa obe strane, dovoljno je da unesemo samo izraz *tcp.port* == 443 (slika 7.3.2.).

🚄 Proba 07092016.pcap								d x
File Edit View Go Capture	Analyze Statistics	Telephony Wireless Tools	Help					
🛋 🔳 🔬 🛞] 🚵 🖾 🔳	٩ @ @ 😤 🛉 .	J 🔲 🗐 Q, Q, Q, 💵						
tcp.port == 443							Expression.	. + тср
No. Time	Source So	urce port Destination	Destination port Ler	ngth Protocol	Info	Expert		^
118 9.828607	91.245.214.1	443 10.0.33.7	54291	117 TLSv1.2	Application			
119 9.828771	91.245.214.1	443 10.0.33.7	54291	60 TCP	443 → 54291	Chat		
120 9.828844	10.0.33.7	54291 91.245.214.1	443	54 TCP	54291 → 443			
121 9.829411	10.0.33.7	54291 91.245.214.1	443	54 TCP	54291 ÷ 443	Chat		
122 9.830409	91.245.214.1	443 10.0.33.7	54291	60 TCP	443 → 54291			
179 12.010183	10.0.33.7	54334 216.58.214.2	443	66 TCP	54334 → 443	Chat		
180 12.023870	216.58.214.2	443 10.0.33.7	54334	66 TCP	443 → 54334	Chat		
181 12.023920	10.0.33.7	54334 216.58.214.2	443	54 TCP	54334 ÷ 443			
182 12.024102	10.0.33.7	54534 216.58.214.2	443	247 ILSv1.2	Client Hello			=
185 12.046156	216.58.214.2	443 10.0.33.7	54334	60 TCP	443 + 54334			_
186 12.046385	216.58.214.2	443 10.0.33.7	54554	1484 11501.2	Server Hello			
187 12.046420	216.58.214.2	443 10.0.33.7	54334	1484 TCP	[ICP segment			
100 12.040454	216 50 214 2	442 10 0 22 7	E440	1200 TLCP	54354 + 445			
 Internet II, St. Asdatte Internet Protocol Version Transmission Control Prot Source Port: 54291 Destination Port: 443 [Stream index: 2] [TCP Segment Len: 0] Sequence number: 1 Acknowledgment number: 	4, Src: 10.0.33. Scol, Src Port: 5 (relative sequence 65 (relative sequence	an 1984 109 109 1, 01 1444 4291 (54291), Dst Port: 44 e number) ack number)	13 (443), Seq: 1,	Ack: 65, Len: 0				
Header Length: 20 byte	i							
Prings: 0x011 (F1R, ACK Window size value: 163 [Calculated window siz [Window size scaling f: [Window size scaling f: Decksum: 0x055a [vali Urgent pointer: 0	83 e: 16333] actor: -1 (unknow Mation disabled]	n)]						
😑 🎽 Proba 07092016						Packets: 748043 · Displayed: 336486 (45.0%) ·	Load time: 0:16.776	Profile: Default
📀 🚞 🕹		💌 煮 🧭				SR 🛓	🔺 🛱 😽 at 🕪 📊	20:54 3.09.2016

Slika 7.3.2. TCP paketi koji se prenose preko porta 443





Wireshark IO Graphs: Proba 07092016

Slika 7.3.3.Broj paketa koji sadrže TCP protokol (žuta linija) u odnosu na ukupan broj paketa (crna linija) u jedinici vremena

Kao što je bilo i za očekivati, TCP paketi čine veći deo ukupnog broja paketa koji se šalje u mreži.

Na slici 7.3.4. je prikazan grafik na kojem su prikazani svi TCP paketi (žuta linija), TCP paketi koji se prenose preko porta 443 (crvena linija) i TCP paketi koji se prenose preko porta 80 (zelena linija).

Wireshark IO Graphs: Proba 07092016



Slika 7.3.4.Poređenje TCP paketa koji se prenose preko porta 443 (crvena linija) i TCP paketa koji se prenose preko porta 80 (zelena linija) u odnosu na ukupan broj TCP paketa (žuta linija)

Desnim klikom na bilo koji TCP paket, otvara se meni u kojem je potrebno kliknuti na opciju Protocol Preferences->Calculate Conversation Timestamps kako bismo mogli da pratimo kašnjenje TCP paketa. U polju *display* filtra možemo uneti izraz *tcp.time_delta>1* i Wireshark će nam prikazati sve pakete kod kojih je kašnjenje veće od jedne sekunde, slika 7.3.5.

📕 Proba 07092	2016.pcap									d X
File Edit Vi	iew Go Capture	e Analyze Stat	istics Telephony	Wireless Tools	Help					
A = 6 0) 🔰 🛅 🔀 🖸	۹ 👄 🔿 🖀	₹ ₹ 🗖 🗖	Q, Q, Q, II						
tcp.time_delt	a>1								Expression	+ TCP
No.	Time	Source	Source port	Destination	Destination port Length	Protocol	Info	Expert		
	38 6.932815	10.0.33.7	5433	9 192.168.2.18	9100	62 TCP	[TCP Retransmission] 54330 → 9100 [SYN.	Note		
	91 7.031858	10.0.33.7	5483	192.168.2.16	8080	62 TCP	[TCP Retransmission] 54331 + 8080 [SYN.	Note		
67	76 20.808591	10.0.33.7	5433	31.24.228.243	443	515 TLSv1.2	Application Data			_
67	78 21.000298	10.0.33.7	5433	5 31.24.228.243	443	930 TLSv1.2	Application Data			
8.	19 22.035451	10.0.33.7	5434	9 192.168.2.16	8080	66 TCP	[TCP Retransmission] 54340 + 8080 [SYN.	. Note		
104	16 26.929692	10.0.33.7		3 192.168.2.18	9100		[TCP Retransmission] 54343 + 9100 [SYN.	Note		
11:	20 28.034749	10.0.33.7	5434	3 192.168.2.16	8080	62. TCP	[TCP Retransmission] 54340 + 8080 [SYN.	. Note		
115	56 28.339830	10.0.33.7	5433	5 31.24.228.243	443	54 TCP	54336 → 443 [FIN, ACK] Seq=324 Ack=330.	. Chat		
115	57 28.339954	10.0.33.7	5433	7 31.24.228.243	443	54 TCP	54337 → 443 [FIN, ACK] Seq=324 Ack=330_	Chat		=
115	58 28.340032	10.0.33.7	5433	3 31.24.228.243	443	54 TCP	54338 → 443 [FIN, ACK] Seq=324 Ack=330_	. Chat		
124	43 32.170230	10.0.33.7	5433	9 31.24.228.243	443	521 TLSv1.2	Application Data			
125	51 32.210605	10.0.33.7	5433	5 31.24.228.243	443	1133 TLSv1.2	Application Data			
130	97 32.932167	10.0.33.7	5434.	3 192.168.2.18	9100	62 TCP	[TCP Retransmission] 54343 + 9100 [SYN.	Note		_
18	76 43.034499	10.0.33.7	5436	1 192.168.2.16	8080	66 TCP	[TCP Retransmission] 54361 + 8080 [SYN_	. Note		
196	54 49.034903	10.0.33.7	5436	1 192.168.2.16	8080	62 TCP	[TCP Retransmission] 54361 + 8080 [SYN.	Note		
204	40 52.928097	10.0.33.7	5436	2 192.168.2.18	9100	66 TCP	[TCP Retransmission] 54362 → 9100 [SYN.			
216	03 57.117318	10.0.33.7	5433	1 216.58.214.2.		55 TCP	[TCP Keep-Alive] 54334 → 443 [ACK] Seq.	Note		
212	24 58.930405	10.0.33.7	5436	2 192.168.2.18	9100	62. TCP	[TCP Retransmission] 54362 + 9100 [SYN.	. Note		
216	58 64.029641	10.0.33.7		5 192.168.2.16	8080	66 TCP	[TCP Retransmission] 54365 + 8080 [SYN.	Note		=
220	93 66.870786	10.0.33.7	5434	1 91.245.214.1.	443	55 TCP	[TCP Keep-Alive] 54341 → 443 [ACK] Seq.	Note		*
Frame 115 Ethernet	7: 54 bytes on	wire (432 bit	s), 54 bytes cap	tured (432 bit	s)	active (concerned)				-
Internet	Protocol Versio	on 4, Src: 10.	0.33.7, Dst: 31.	24.228.243	11e_bc.bc.cd (50.17.					
4 Transmiss	ion Control Pro	otocol, Src Po	rt: 54337 (54337), Dst Port: 4	43 (443), Seq: 324,)	Ack: 3301, Len: 0				
Source	Port: 54337									=
Destina	ation Port: 443	3								
[Stream	m index: 9]									
[TCP S	egment Len: 0]									
Sequen	ce number: 324	(relative :	sequence number)							
Acknow	ledgment number	-: 3301 (re.	lative ack numbe	r)						
Header	Length: 20 byt	tes								
Flags:	0x011 (FIN, AC	K)								
Window	size value: 16	5360								
[Cales	Inted window of	Sec. 654401								*
🔵 🎽 Flags	(12 bits) (tcp.flags),	2 bytes					Packets: 748043 • Displayed: 16	35 (0.2%) · Load 1	ime: 0:13.931 P	rofile: Default
()	3 🕘							EN 🔺 🦽	🕻 🕼 😹 🕪 📊	21:31 3.09.2016

Slika 7.3.5.TCP paketi kod kojih je kašnjenje veće od jedne sekunde

Na slici 7.3.6. je prikazan grafik koji nam maksimalno, minimalno i prosečno kašnjenja paketa. Ovaj grafik smo dobili korišćenjem MAX (*), MIN (*) i AVG (*) funkcije za y-osu uz filtar *tcp.time_delta*.



Slika 7.3.6.Kašnjenja pojedinih TCP paketa

Pošto nam TCP paketi koji prenose ACK potvrdu o resetu ili završetku TCP konekcije, nisu toliko od interesa, unećemo složeniji izraz tcp.time_delta>1 and tcp.flags.fin==00 and tcp.flags.reset==0 koji će nam prikazati pakete koji imaju veća kašnjenja, a mogu biti zanimljivi za analizu i otkrivanje problema u mreži (slika 7.3.7.).



Slika 7.3.7. TCP paketi kod kojih je kašnjenje veće od jedne sekunde a ne nose ACK potvrdu o resetu ili završetku TCP konekcije

Paketi koje Wireshark prikazuje u glavnom prozoru nakon filtriranja na osnovu izraza $tcp.time_delta>1$ and tcp.flags.fin==00 and tcp.flags.reset==0 uglavnom predstavljaju pakete koji se ponovo šalju (*retransmission*). Ako želimo da vidimo pakete koji su ponovo poslati, dovoljno je da u polje *display* filtra unesemo tcp.analysis.retransmission.

Wireshark IO Graphs: Proba 07092016



Slika 7.3.8. Broj frejmova koji se ponovo šalje kroz TCP protokol

Broj frejmova koji se ponovo šalju kroz TCP protokol može se dobiti i grafički, ako izaberemo funkciju COUNT FRAMES (*) za y-osu, a filtar *tcp.analysis.retransmission* (slika 7.3.8.).

Problematične TCP pakete možemo filtrirati i unošenjem izraza *tcp.analysis.flags* && *!tcp.analysis.window_update*. Grafik problematičnih TCP paketa se može videti na slici 7.3.9.



Wireshark IO Graphs: Proba 07092016



TCP pakete možemo filtrirati i na osnovu sadržaja samog paketa korišćenjem izraza *tcp* contains. Ako želimo da vidimo koji TCP paketi imaju setovane kontrolne bite SYN, ACK, FIN, možemo da ih filtritramo na osnovu izraza *tcp.flags.syn*, *tcp.flags.ack* i *tcp.flags.fin*, respektivno.

7.4. Analiza paketa UDP protokola

Za razliku od TCP protokola koji nam nudi različite mogućnosti za filtriranje, UDP protokol nam pruža manje opcija zbog jednostavnosti samog zaglavlja. UDP možemo filtrirati na osnovu porta koji se koristi za komunikaciju, bilo da je u pitanju izvorišni ili odredišni port. UDP pakete možemo filtrirati i na osnovu veličine paketa u vidu izraza *udp.length* == 48. Ako pogledamo prikazane pakete čija je veličina 48, primetićemo da su to samo paketi koji pripadaju protokolu QUIC (*Quick UDP Internet Connections*). UDP pakete možemo filtrirati i na osnovu kontrolne sume – izraz *udp.checksum_bad* prikazaće sve pakete kod kojih kontrolna suma ne pokazuje dobru vrednost.

Na slici 7.4.1. prikazan je broj UDP paketa u jedinici vremena (crvenom bojom) u odnosu na ukupan broj paketa (crnom bojom) u snimljenom fajlu.

Wireshark IO Graphs: Proba 07092016



Slika 7.4.1.Broj paketa koji sadrže UDP protokol (crvena boja) u odnosu na ukupan broj paketa (crna boja) u jedinici vremena

Ako želimo da vidimo samo pakete koji sadrže UDP protokol, dovoljno je da u polje *display* filtra unesemo *udp* i dobićemo ispis na glavnom prozoru kao na slici 7.4.2.

📕 Proba 07092016.pcap						9 X
File Edit View Go Capture Analyze Statistics Teleph	ony Wireless Tools Help					
▲ ■ △ ④ ▲ □ X ■ 4 ← ∞ ∞ ∓ 4 □						
					r) Expression	+ TCP
No. Time Source Source port	t Destination Destina	tion port Le	angth Protocol	Info	Expert	*
4639 198.115177 212.200.190.166	53 10.0.33.7	55352	160 DNS	Standard query response 0x02b6 A api4.adsflow.net A 31.24		
4640 198.115602 212.200.190.166	53 10.0.33.7	59504	545 DNS	Standard query response 0x58e1 A pixel.quantserve.com CNAM.		
4650 198.137098 212.200.190.166	53 10.0.33.7	52557	491 DNS	Standard query response 0xfb16 A synchroscript.deliveryeng.		
4652 198.138151 10.0.33.7 640	17 212.200.190.166	53	83 DNS	Standard guery 0x978b A b.scorecardresearch.com		
4653 198.139248 212.200.190.166	53 10.0.33.7	64017	533 DNS	Standard query response 0x978b A b.scorecardresearch.com C.		
4687 198.175892 10.0.33.7 611	02 212,200,190,166	53	76 DNS	Standard query 0xb7d3 A www.facebook.com		
4689 198.177803 10.0.33.7 592	61 212,200,190,166	53	81 DNS	Standard query 0x8ceb A staticxx.facebook.com		
4694 198.177853 212.200.190.166	53 10.0.33.7	61102	244 DNS	Standard query response 0xb7d3 A www.facebook.com CNAME st.		
4697 198,179466 212,200,190,166	53 10.0.33.7	59261	255 DNS	Standard query response Øx8ceb A staticxx.facebook.com CNA.		
4711 198, 196804 188, 120, 127, 102 4	43 10.0.33.7	64721	79 OUIC	Pavload (Encrypted), CID: 12279810855815576888, Sec: 5		
4756 198.225955 10.0.33.7 624	04 212.200.190.166	53	85 DNS	Standard guery 0x1b64 A entitlements.jwplayer.com		
4802 198.259187 212.200.190.166	53 10.0.33.7	62404	414 DNS	Standard query response 0x1b64 A entitlements.jwplayer.com.		
4816 198.273944 10.0.32.165 330	14 239.255.255.250	1900	167 SSDP	M-SEARCH * HTTP/1.1	Chat	
4818 198.293552 10.0.33.7 653	34 212.200.190.166	53	79 DNS	Standard query 0x187a A accounts.google.com		
4823 198.300189 212.200.190.166	53 10.0.33.7	65334	231 DNS	Standard query response 0x187a A accounts.google.com A 216		
4827 198.318472 10.0.33.7 653	35 216.58.209.205	443	1392 QUIC	Client Hello, CID: 8243828842489949717, Seq: 1	Note	
4829 198.319093 10.0.33.7 653	35 216.58.209.205	443	758 OUIC	Payload (Encrypted), CID: 8243828842489949717, Seg: 2		
4838 198.333966 10.0.33.7 653	35 216.58.209.205	443	1392 OUIC	Client Hello, CID: 8243828842489949717, Seg: 3	Note	
4839 198.335455 216.58.209.205 4	43 10.0.33.7	65335	1392 QUIC	Payload (Encrypted), Seq: 1		+
 Frame 4711: 79 bytes on wire (632 bits), 79 bytes Ethernet II, Snc: HuaweiTe_bc:6c:d (90:12/actbc Internet Protocol Version 4, Src: 188.120.127.10 User Datagram Protocol, Src Port: 443 (443), Dst 4 QUTC (Quick UPD Internet Connections) 	s captured (632 bits) :6c:cd), Dst: AsustekC_9 2, Dst: 10.0.33.7 Port: 64721 (64721)	a:d0:88 (78	3:24:af:9a:d0:88)			
Public Flags: 0x0c CID: 12279810855815576888 Sequence: 5 Payload: 9a65043b83e90a4c7f81ebbcc738cb6b6ac50	34882bbbc7ff					
0000 78 24 af 9a d8 89 91 7 ac bc 6c cd 08 04 09 04 05 04 05 01	5 00 x\$1E. 5 00 .A.@.4					
🕘 🌋 User Datagram Protocol: Protocol				Packets: 748043 • Displayed: 76599 (10.2%) • Load time: 0:12	2.387 Profile	le: Default
📀 📜 🕘 🙆 📉				SR 🔺 🔒 📴 🎀	0:0 14.09 في	07

Slika 7.4.2. Paketi koji sadrže UDP zaglavlje

Korišćenjem I/O Graph opcije koju nudi Wireshark, možemo da uporedimo broj paketa u jedinici vremena za UDP pakete koji koriste port 137 i UDP pakete koji koriste port 443 (slika 7.4.3.).

Wireshark IO Graphs: Proba 07092016



Slika 7.4.3.Poređenje UDP paketa koji se prenose preko porta 443 (zelene tačkice) i UDP paketa koji se prenose preko porta 137 (žute tačkice) u odnosu na ukupan broj UDP paketa (crvena linija)

Grafici se dobijaju unošenjem odgovarajućih filtara u polja I/O Graph dela Wireshark aplikacije udp.port==443 i udp.port==137. Filtar možemo napisati za bilo koji port koji nam je od interesa za analizu ili za bilo koji port preko koga se prenosi najveći broj UDP paketa.

Zanimljivo za analizu bi moglo da bude poređenje paketa koji sadrže UDP protokol sa paketima koji sadrže TCP protokol.



Wireshark IO Graphs: Proba 07092016

Slika 7.4.4.Poređenje UDP i TCP paketa u jedinici vremena

Na slici 7.4.4. je prikazan grafik na kojem možemo da vidimo da imamo znatno više TCP paketa (plava linija) u jedinici vremena nego što je to slučaj sa UDP paketima (crvena linija).

Ako želimo da vidimo UDP pakete koji se prenose preko porta 53, dovoljno je da u polju display filtra unesemo udp.dstport == 53 i Wireshark će nam izlistati sve pakete koji se šalju DNS serveru. Iste te pakete možemo i da filtriramo na osnovu IP adrese DNS servera na sledeći način ip.dst == 212.200.190.166 and udp. Na slici 7.4.5. su prikazani paketi koji zadovoljavaju pomenuti filtar.

Proba 07092016.pcap				
File Edit View Go Capture Analyze Statistics	s Telephony Wireless Tools	Help		
📶 🔳 🖉 😣 🛄 🗁 🗙 💆 🔍 🗢 🗢 🕾 🖗	1			
p.dst == 212.200.190.166 and udp				Expression + TOP
No. Time Source 5	Source port Destination [Destination port Length Protoco	I Info Expert	
4990 199.294886 10.0.33.7	55028 212.200.190.166	53 90 DNS	Standard query 0xeb3e A securepubads.g	
5072 199.612761 10.0.33.7	59276 212.200.190.166	53 89 DNS	Standard query 0xf06e A pagead2.google	
5074 199.618808 10.0.33.7	54251 212.200.190.166	53 85 DNS	Standard query 0x398b A tpc.googlesynd…	
5076 199.621305 10.0.33.7	52144 212.200.190.166	53 76 DNS	Standard query 0xe99e A ads.pubmatic.c	
5079 199.642627 10.0.33.7	56333 212.200.190.166	53 84 DNS	Standard query 0xd393 A www.googleadse…	
5100 199.671264 10.0.33.7	57651 212.200.190.166	53 87 DNS	Standard query 0xb6d0 A googleads.g.do	
5101 199.672463 10.0.33.7	59439 212.200.190.166	53 80 DNS	Standard query 0x3d0c A cm.g.doublecli	
5211 199.800087 10.0.33.7	56369 212.200.190.166	53 80 DNS	Standard query 0xf485 A showads.pubmat	
5221 199.834771 10.0.33.7	52483 212.200.190.166	53 80 DNS	Standard query 0x465d A aep.emea.mxpti	
5222 199.834771 10.0.33.7	53756 212.200.190.166	53 79 DNS	Standard query 0xb228 A image6.pubmati	
5285 199.945965 10.0.33.7	65327 212.200.190.166	53 76 DNS	Standard query 0xb7c1 A idsync.rlcdn.c	
5286 199.947044 10.0.33.7	55859 212.200.190.166	53 /1 DNS	Standard query 0xac2e A ad.turn.com	
5287 199.948596 10.0.33.7	60157 212.200.190.166	53 72 DNS	Standard query 0x3e33 A 1b.adnxs.com	
5288 199.949826 10.0.33.7	52041 212.200.190.166	53 77 DNS	Standard query 0x0et9 A pm-m.d.chango	
5332 200.055605 10.0.33.7	59194 212.200.190.166	53 /9 DNS	Standard query 0x9948 A image2.pubmati	
T* 536/ 200.169963 10.0.33.7	63021 212.200.190.166	53 80 DNS	Standard query 0x01a7 A aktrack.pubmat	
5512 200.935755 10.0.35.7	49089 212.200.190.106	55 80 DNS	Standard query exceed A platform.twitt.	
Frame 5367: 80 bytes on wire (640 bits),	80 bytes captured (640 bits	s)		*
Ethernet II, Src: AsustekC_9a:d0:88 (78:2)	4:af:9a:d0:88), Dst: Huawe:	iTe_bc:6c:cd (90:17:ac:bc:	6c:cd)	
D Internet Protocol Version 4, Src: 10.0.33	3.7, Dst: 212.200.190.166			
4 User Datagram Protocol, Src Port: 63021 (63021), Dst Port: 53 (53)			E
Source Port: 63021				
Destination Port: 53				
Length: 46				
Checksum: 0x6c5c [Validation disabled]				
[Stream index: 468]				
 Domain Name System (query) 				
[Response In: 5368]				•
0000 90 17 ac bc 6c cd 78 24 af 9a d0 88	08 00 45 001.×\$	E.		
0010 00 42 57 5c 00 00 80 11 24 d9 0a 00	21 07 d4 c8 .BW\ \$			
0020 be a6 f6 2d 00 35 00 2e 6c 5c 01 a7	01 00 00 01			
0030 00 00 00 00 00 00 07 61 6b 74 72 61	63 65 68 70a ktra	ack.p		
0040 75 62 6d 61 74 69 63 03 63 6T 6d 00	00 01 00 01 UDMatic. com.			
Z Destination Part (udp.dstpart), 2 bytes			Packets: 748043 * 0	Displayed: 344 (0.0%) + Load time: 0:13.212 Profile: Default
			10000017100101	1.50
	😬 🔛 🔼			SR 🔺 🏦 🔐 🍡 📣 1158 14.09.2016

Slika 7.4.5. Pretraga UDP paketa koji imaju odredišnu adresu DNS servera

7.5. Analiza paketa ICMP protokola

Ako želimo da vidimo samo pakete koji sadrže ICMP protokol, dovoljno je da u polje *display* filtra unesemo *icmp* i dobićemo ispis na glavnom prozoru kao na slici 7.5.1.

Proba 07092016.pcap				- ē 🔀
File Edit View Go Capture Analyze Statis	tics Telephony Wireless Tools Help			
▲ ■ ∅ ⊕ 🔰 🗅 🗙 🖄 ۹ ⇔ 🕾	₮ 业 🗐 🗐 @, @, @,			
icmp				Expression + TCP
No. Time Source	Source port Destination Destination port	Length Protocol Info	Expert	
2280 71.683194 10.0.33.7	8.8.8	74 ICMP Echo (ping) request	id=0x0001, seq=1/	
2281 71.692190 8.8.8.8	10.0.33.7	74 ICMP Echo (ping) reply	id=0x0001, seq=1/	
2295 72.684335 10.0.33.7	8.8.8.8	74 ICMP Echo (ping) request	id=0x0001, seq=2/	
2296 72.694322 8.8.8.8	10.0.33.7	74 ICMP Echo (ping) reply	id=0x0001, seq=2/	
2310 73.685220 10.0.33.7	8.8.8	74 ICMP Echo (ping) request	id=0x0001, seq=3/	
2311 73.694428 8.8.8.8	10.0.33.7	74 ICMP Echo (ping) reply	id=0x0001, seq=3/	
2336 74.686274 10.0.33.7	8.8.8.8	74 ICMP Echo (ping) request	id=0x0001, seq=4/	5
2337 74.695255 8.8.8.8	10.0.33.7	74 ICMP Echo (ping) reply	id=0x0001, seq=4/	
2911 115.093693 10.0.33.7	93.87.27.51	74 ICMP Echo (ping) request	id=0x0001, seq=5/ Warn	
2962 119.754558 10.0.33.7	93.87.27.51	74 ICMP Echo (ping) request	id=0x0001, seq=6/ Warn	
3008 124.753719 10.0.33.7	93.87.27.51	74 ICMP Echo (ping) request	id=0x0001, seq=7/ Warn	
3069 129.754027 10.0.33.7	93.87.27.51	74 ICMP Echo (ping) request	id=0x0001, seq=8/ Warn	
3251 146.691203 10.0.33.7	10.2.0.25	74 ICMP Echo (ping) request	id=0x0001, seq=9/… Warn	
3318 151.255039 10.0.33.7	10.2.0.25	74 ICMP Echo (ping) request	id=0x0001, seq=10 Warn	
3358 156.255522 10.0.33.7	10.2.0.25	74 ICMP Echo (ping) request	id=0x0001, seq=11 Warn	
3484 161 253688 18 8 33 7	10 2 0 25	24 TCMP Echo (ning) ceques	id-8x8881 sec-12 Marco	
Ethernet II, Src: AsustekC_9a:d0:88 (78)	8:24:af:9a:d0:88), Dst: HuaweiTe_bc:6c:cd	(90:17:ac:bc:6c:cd)		A
Internet Protocol Version 4, Src: 10.0.	.33.7, Dst: 10.2.0.25			
Internet Control Message Protocol				
Type: 8 (Echo (ping) request)				
Code: 0				
Checksum: 0x4d4f [correct]				
Identifier (BE): 1 (0x0001)				E
Identifier (LE): 256 (0x0100)				
Sequence number (BE): 12 (0x000c)				
Sequence number (LE): 3072 (0x0c00)				
[No response seen]				
Data (32 bytes)				-
0000 90 17 ac bc 6c cd 78 24 af 9a d0 8	38 08 00 45 00l.x\$F.			
0010 00 3c 54 8d 00 00 80 01 b1 12 0a 0	30 21 07 0a 02 . <t< td=""><td></td><td></td><td></td></t<>			
0020 00 19 08 00 4d 4f 00 01 00 0c 61 0	52 63 64 65 66MOabcdef			
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 7	72 73 74 75 76 ghijklmn opqrstuv			
0040 77 61 62 63 64 65 66 67 68 69	wabcdefg hi			
Internet Control Message Protocol: Protocol			Packets: 748043 • Displayed: 16 (0	.0%) · Load time: 0:11.540 Profile: Default
			11	
🐨 🔲 🔮 🚺 🖄				SR 🔺 🏦 📴 🎼 📬 🕪 2302 14.09.2016
		T I I I I I I I I I I		

Slika 7.5.1. Paketi koji sadrže ICMP zaglavlje

Na slici 7.5.2. prikazan je broj ICMP paketa u jedinici vremena (crvenom bojom) u odnosu na ukupan broj paketa (crnom bojom) u snimljenom fajlu.



Wireshark IO Graphs: Proba 07092016

Slika 7.5.2.Broj paketa koji sadrže ICMP protokol (crvena boja) u odnosu na ukupan broj paketa (crna boja)

Možemo primetiti da je broj ICMP paketa znatno manji u odnosu na sve pakete, a javljaju se na početku snimanja paketa.

Ako u polje *display* filtra unesemo *icmp.type* == 8, kao rezultat u glavnom prozoru videćemo samo ICMP pakete koji u sebi sadrže zahtev (*request*). Izraz *icmp.type* == 0 kao rezultat prikazuje samo ICMP pakete koji u sebi sadrže odgovor (*reply*).

Wireshark IO Graphs: Proba 07092016



Slika 7.5.3.Broj paketa koji sadrže ICMP zahtev (žuta boja) i broj paketa koji sadrže ICMP odgovor (zelena boja) u jedinici vremena

Ako nas zanima da li je u toku snimanja bilo nedostupnih hostova, u polje display filtra unećemo*icmp.type* == 3 && icmp.code == 3.

ICMP pakete možemo filtrirati na osnovu različitih vrednosti polja type i code u ICMP paketu.

7.6. Analiza paketa DHCP protokola

Ako želimo da vidimo samo pakete koji sadrže DHCP protokol, dovoljno je da u polje *display* filtra unesemo *bootp* (ranija verzija protokola za dodeljivanje adresa koji odgovara današnjem DHCP protokolu) i dobićemo ispis na glavnom prozoru kao na slici 7.6.1.

📕 Pr	oba 07092016.pcap								- d <u>×</u>
File	Edit View Go	Capture Analyze S	Statistics Telephony Wireless Tools	Help					
4	1 1 1 🛞 🔒 🗅	X C 9 0 0	≝ 7 & □ ■ Q Q Q W						
bo	atra								TCP
	τ	0.000	and a set of the set o	0	Level Destand			Count -	
NO.	1me	50urce	Source port Destination	Destination port	Length Protocol	DICD Tafaan	Terresting TD 0050044	Expert	- I
	889 23.210445	10.0.32.18/	68 255.255.255.255	67	342 DHCP	DHCP Inform	- Transaction ID 0x52284	han .	
	1951 41 455005	10.0.33.2	00 200.200.200.200	67	342 DHCP	DHCP Inform	Transaction ID 0x40150	her	=
	1051 41.455905	10.0.22.1	60 200.200.200.200	67	347 DHCP	DHCP Request	- Transaction ID 0x0976		
-	2316 73 743415	10.0.32.49	68 255 255 255 255	67	342 DHCP	DHCP Ack	- Transaction ID 0x86637	•••	
	2013 77 596589	10.0.32.49	68 255 255 255 255	67	342 DHCP	DHCP Inform	- Transaction ID 0v7b123		
	3353 155 84753	3 10 0 32 49	68 255 255 255 255	67	342 DHCP	DHCP Inform	- Transaction ID Gyac547	**	
	3439 164 79939	60000	68 255 255 255 255	67	342 DHCP	DHCP Discover	- Transaction ID 0x417d7	**	
	3481 166 91476	4 8 8 8 8	68 255 255 255 255	67	342 DHCP	DHCP Request	- Transaction ID 0x417d7	***	
	3522 168 78902	80000	68 255 255 255 255	67	356 DHCP	DHCP Discover	- Transaction ID 0x41/4/		
	3560 160 81330	78888	68 255 255 255 255	67	368 DHCP	DHCP Request	- Transaction ID 0x316fg	***	
	6300 203 69280	9 10 0 33 7	68 255 255 255 255	67	342 DHCP	DHCP Inform	- Transaction TD 0x37cd4		
	6301 203 69337	3 10 0 32 1	67 10 0 33 7	68	350 DHCP	DHCP ACK	- Transaction ID 0x37cd4	•••	
	12125 257 42343	7 10 0 33 108	68 255 255 255 255	67	342 DHCP	DHCP Inform	- Transaction ID 0x7e51f		
	13777 264 94490	9 10 0 33 6	68 255 255 255 255	67	342 DHCP	DHCP Inform	- Transaction ID 0x857d3		
	16621 289 64878	7 10 0 32 78	62 255 255 255 255	67	B42 DHCP	DHCP Inform	- Transaction TD 0x05858		
	Length: 316								
Þ	Checksum: 0xc79	9b [validation dis	abled						
	[Stream index:	114]	2						
4 Bo	otstrap Protoco	1 (ACK)							
	Message type: 8	Boot Reply (2)							7
	Hardware type:	Ethernet (0x01)							
	Hardware addres	ss length: 6							
	Hops: 0								
	Transaction ID:	0x6976b6e5							
	Seconds elapsed	1: 0							
⊳	Bootp flags: 0x	k0000 (Unicast)							
	Client IP addre	ess: 0.0.0.0							
	Your (client) 1	TP address · 10 0 3	3.4						
0000	00 03 aa fe 59	9 82 90 17 ac bc	6c cd 08 00 45 00Y1	E.					
0010	01 50 96 80 00	0 00 ff 11 cf 17	0a 00 20 01 0a 00 .P						1
0020	21 04 00 43 00	0 44 01 3c c7 9b	02 01 06 00 69 76 !C.D.<	iv					
0030	00 00 00 00 00 00	5 070 010 010 010 010 010 0 3 000 000 03 aa fa	50 87 88 88 88 88 88						
0050	00 00 00 00 00	00 00 00 00 00 00	00 00 00 00 00 00						
•	Registeren Bratar	al Protocol					Dacketer	749042 - Displayed: 47 (0.0%) - Load time: 0:11 45	E Droßley Defaul
	Buotatrap Protoco			-			Packets:	740045 * Displayee: 47 (0.0%) * Load time: 0:11.45	> Profile: Default
-		9 🙆 2	🔄 😬 🔟 🔟					SR 🔺 🕰 🛱 🎼	2:35 14.09.2016

Slika 7.6.1. Paketi koji sadrže DHCP zaglavlje

Na slici 7.6.2. prikazan je broj DHCP paketa u jedinici vremena (crvenom bojom) u odnosu na ukupan broj paketa (crnom bojom) u snimljenom fajlu.

Ako želimo da vidimo pakete koji sadrže DHCP protokol, razmenjeni između dve IP adrese (10.0.32.1. i 10.0.33.7), potrebno je da filtriramo pakete sledećim izrazom (*ip.addr* == 10.0.32.1) && (*ip.addr* == 10.0.33.7) and bootp.

Možemo napraviti *display* filtar na osnovu bilo kog polja DHCP zaglavlja, na osnovu MAC i IP adrese klijenta, tipa poruke, broja hopova, bilo kog sadržaja opcionog polja – dovoljno je da kliknemo desnim klikom na taj red DHCP zaglavlja prikazan u glavnom prozoru i izaberemo opciju *Apply as Filter*. Možemo i da kombinujemo izraz za *display* filtar, na osnovu vrednosti više polja DHCP paketa, tada koristimo opciju *Prepare a Filter*, tako da imamo mogućnost da proširimo izraz a da se filtriranje ne pokrene automatski kao kod opcije *Apply as Filter*.

Ako unesemo izraz *bootp.option.domain_name_server* == 212.200.190.166 u polje za *display* filtar, Wireshark će nam prikazati samo pakete kod kojih je upisana odgovarajuća IP adresa DNS servera.
Wireshark IO Graphs: Proba 07092016



Slika 7.6.2.Broj paketa koji sadrže DHCP protokol (crvena boja) u odnosu na ukupan broj paketa (crna boja)

Izraz *bootp.id* == 0x037cd47a će nam pokazati pakete koji imaju odgovarajući *transaction* ID i na taj način možemo pratiti kako teče komunikacija.

Možemo i da filtriramo sve DHCP *discover* pakete sledećim izrazom *bootp.option.dhcp* == 1. Dobijanje IP adrese odvija se kroz DORA proces (razmenu *Discover*, *Offer*, *Request* i *ACK*paketa).

Moglo bi da bude zanimljivo da prikažemo koliko se paketa koji imaju DHCP zaglavlje prenosi UDP protokolom. Na slici 7.6.3. je prikazan grafik koji pokazuje broj paketa u jedinici vremena koji sadrže UDP zaglavlje (crvene tačke) i broj paketa u jedinici vremena koji sadrže DHCP zaglavlje (plave tačke). Možemo da primetimo da se većina DHCP paketa prenosi preko UDP protokola.

Wireshark IO Graphs: Proba 07092016



Slika 7.6.3.Broj paketa koji sadrže DHCP protokol (crvena linija) u odnosu na pakete koji sadrže UDP protokol (zelena linija)

7.7. Analiza paketa DNS protokola

Ako želimo da vidimo samo pakete koji sadrže DNS protokol, dovoljno je da u polje *display* filtra unesemo*dns* i dobićemo ispis na glavnom prozoru kao na slici 7.7.1.

4	Proba 07092016.pcap							- P	×
File	Edit View Go Capture	Analyze Statistics	Telephony Wireless Tools Help						
A.	🔲 🔬 🙃 🔰 💿 🖄 🖻	९ 👄 🕾 🐴	୬ 🚍 🗏 ୧. ୧. ୧. ୩						
	Ins						Expression		TCP
No.	Time	Source So	urce port Destination	Destination port Length	Protocol	Info	Expert		
	619512 761.256211	212.200.190	53 10.0.33.7	50277	474 DNS	Standard query response Øxddeb A ssl.gstatic.co	m		
	619723 764.273402	10.0.33.7	57660 212.200.190.166	53	69 DNS	Standard query 0x7806 A gmail.com			
	619724 764.274472	212.200.190	53 10.0.33.7	57660	228 DNS	Standard query response 0x7806 A gmail.com A 21	6		
	619751 764.363030	10.0.33.7	63402 212.200.190.166	53	75 DNS	Standard query 0x44c3 A mail.google.com			
	619752 764.364095	212.200.190	53 10.0.33.7	63402	254 DNS	Standard query response 0x44c3 A mail.google.co	m		
	619780 764.471899	10.0.33.7	62544 212.200.190.166	53	79 DNS	Standard query 0x8c41 A accounts.google.com			
	619782 764.474375	212.200.190	53 10.0.33.7	62544	231 DNS	Standard query response 0x8c41 A accounts.googl	e		
	619901 764.770280	10.0.33.7	61254 212.200.190.166	53	80 DNS	Standard query 0xdc5b A accounts.youtube.com			
	619902 764.771292	212.200.190	53 10.0.33.7	61254	260 DNS	Standard query response 0xdc5b A accounts.youtu	b		
	619946 765.036305	10.0.33.7	51453 212.200.190.166	53	79 DNS	Standard query 0x529b A clients1.google.com			-
	619947 765.037324	212.200.190	53 10.0.33.7	51453	255 DNS	Standard query response 0x529b A clients1.googl	e		
	620674 769.753157	10.0.33.7	52221 212.200.190.166	53	75 DNS	Standard query 0x9f73 A www.youtube.com			-
4 [Destination Port: 551 Length: 259 4 Checksum: 8xa87a [val [Good Checksum: Fal [Stream Index: 1403] [Ineruest Int 620715] [Time: 0.001062000 Eee Transaction ID: 0x4769 Flags: 0x48180 Standar Questions: 1 Answer RRs: 2 Authority RRs: 4	74 idation disabled] Lse] onnse) conds] c d query response,	No error						E
000 001 002 003 004 005	0 70 24 41 93 00 88 89 0 01 03 88 1d 00 00 90 99 0 11 03 88 1d 00 00 90 99 0 21 07 00 35 d7 86 00 0 00 02 00 04 00 04 03 0 65 75 73 65 72 63 61 0 00 00 01 00 01 c0 cc	17 at bc bc cd 0 11 7a 56 d4 c8 b ef a8 7a 4f 0c 8 6c 68 33 11 67 6 6e 74 65 6e 74 0 00 05 00 01 00 1	0 00 00 20 20 111. 111. 1211. 121. 1211. 121. 1211. 1211. 121.1. 121.1. 121.1. 121.1. 121.1. 121.1. 121.1. 121.1. 121.1. 121.1. 121.1. 121.1. 121.11. 121.1. 121.11. 121.1. 121.11. 121.11. 121.1.1. 121.1.1. 121.1.1. 121.1.1. 121.1.1. 121.1.1. 121.1.1. 121.1.1.1. 121.1.1.1. 121.1.1.1.1. 121.1.1.1.1.1.1. 121.1.1.1.1.1.1. 121.1.1.1.1.1.1.1.1. 121.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.						-
0	Z Domain Name System: Prob	ocol				Packets: 748043 * Displayed: 688 (0.1%) * Load time: 0:	15.223	Profile:	Default
6) 👸 🕘 🛛		🥖 🕂 🕺			SR 🔺 🎪 🛱	🔞 🛋 🚯	20:03 14.09.20	3 016

Slika 7.7.1. Paketi koji sadrže DNS zaglavlje

Na slici 7.7.2. prikazan je broj DNS paketa u jedinici vremena (crvenom bojom) u odnosu na ukupan broj paketa (crnom bojom) u snimljenom fajlu.

Wireshark IO Graphs: Proba 07092016



Slika 7.7.2. Broj paketa koji sadrže DNS protokol (crvena boja) u odnosu na ukupan broj paketa (crna boja)

Možemo napraviti *display* filtar na osnovu bilo kog polja DNS zaglavlja, kao što su *transaction* ID, različiti tipovi zastava (*flags*), na osnovu DNS upita i odgovora – dovoljno je da kliknemo desnim klikom na taj red DNS zaglavlja prikazan u glavnom prozoru i izaberemo opciju *Apply as Filter*.

Moglo bi da bude zanimljivo da prikažemo koliko se paketa koji imaju DNS zaglavlje prenosi UDP protokolom. Na slici 7.7.3. je prikazan grafik koji pokazuje broj paketa u jedinici vremena koji sadrže UDP zaglavlje (zelena linija) i broj paketa u jedinici vremena koji sadrže DNS zaglavlje (crvena linija).



Wireshark IO Graphs: Proba 07092016

Slika 7.7.3.Broj paketa koji sadrže DNS protokol (crvena linija) u odnosu na pakete koji sadrže UDP protokol (zelena linija)

Ako u polje *display* filtra unesemo izraz *dns.flags.response* == 0, kao rezultat dobićemo listu paketa koji predstavljaju standardan DNS upit (*query*).

Izrazom (*ip.addr* == 10.0.33.7) && (*ip.addr* == 212.200.190.166) and dnsfiltiramo sve pakete koji su razmenjeni između nas i DNS servera.

Ako u polje *display* filtra unesemo izraz dns.id == 0x8c41, Wireshark će prikazati sve DNS pakete koji imaju pomenuti *transaction*ID. Na taj način možemo da pratimo kako teče komunikacija.

Ako kliknemo na opciju Statistics->DNS u meniju dobićemo prozor prikazan na slici 7.7.4.

Na osnovu vrednosti za broj paketa, možemo da zaključimo da je pet puta veći broj paketa za DNS upit u odnosu na DNS odgovor.

Wireshark • DNS • Proba 07092016									- 0
pic / Item	Count	t Average	Min val Max	al Rate (ms)	Percent	Burst rate	Burst start		
Total Packets	1922			0.0021	100%	0.5700	403.370		
4 rcode	1922			0.0021	100.00%	0.5700	403.370		
No such name	2			0.0000	0.10%	0.0200	405,908		
No error	1920			0.0021	99,90%	0.5700	403,370		
4 opcodes	1922			0.0021	100.00%	0.5700	403.370		
Standard query	1922			0.0021	100.00%	0.5700	403,370		
Query/Response	1922			0.0021	100.00%	0.5700	403.370		
Response	463			0.0005	24.09%	0.2800	403,372		
Query	1459			0.0016	75.91%	0.2900	403.370		
 Ouery Type 	1922			0.0021	100.00%	0.5700	403,370		
Unused	122			0.0001	6.35%	0.1600	169.240		
TXT (Text strings)	3			0.0000	0.16%	0.0100	169.439		
SRV (Server Selection)	12			0.0000	0.62%	0.0100	289.446		
PTR (domain name PoinTeR)	113			0.0001	5.88%	0.0600	839.610		
ΔΔΔΔ (IPv6 Address)	210			0.0002	10.93%	0.1200	343.721		
A (Host Address)	1404			0.0015	73.05%	0.5700	403 370		
* (A request for all records the server/cache has available)	58			0.0001	3.02%	0.0400	832 211		
d Class	1922			0.0021	100.00%	0.5700	403.370		
Linknown class (32769)	27			0.0000	1 40%	0.0200	414 278		
Unknown class (0)	122			0.0001	6.35%	0.1600	169.240		
IN IN	1773			0.0019	92.25%	0.5700	403 370		
Response Stats	0			0.0000	100%	-			
no of questions	463	0.74	0 1	0.0005	10010	0.2800	403 372		
no of authorities	462	2.05	0 12	0.0005		0.2800	402 272		
no. of appliant	462	410	0 21	0.0005		0.2000	403.372		
no. of additionals	463	3.07	0 16	0.0005		0.2800	403.372		
Duary Chate	0	5.57	0 10	0.0000	100%	0.2000	405.572		
Oname lan	1450	15.06	0 77	0.0016	100 %	0.2000	402 270		
4 Jahol State	0	10,00	v 11	0.0000		0.2500	405.570		
(th level or more	126			0.0001		0.0700	205 202		
2rd Level of Hore	227			0.0001		0.2700	402 270		
2nd Level	41			0.0000		0.0600	403.578		
Int Level	41			0.0000		0.0000	405.576		
Tst Level	1022	121.06	20 1440	0.0071	1008	0.2400	402 270		
Payload size	1922	121.00	20 1440	0.0021	100 %	0.3700	405.570		
av filter: Enter a display filter									Apph
								Copy Save	as Close
	al			_		_			4:38
/ 🚍 🤍 🔛 🙇 📖	00								15.09.2

Slika 7.7.4. Ukupan broj DNS paketa, kao i broj DNS paketa sa različitim kodovima

7.8. Analiza paketa HTTP protokola

Ako želimo da vidimo samo pakete koji sadrže HTTP protokol, dovoljno je da u polje *display* filtra unesemo*http* i dobićemo ispis na glavnom prozoru kao na slici 7.8.1.

Možemo napraviti *display* filtar na osnovu bilo kog polja HTTP zaglavlja, dovoljno je da kliknemo desnim klikom na taj red HTTP zaglavlja prikazan u glavnom prozoru i izaberemo opciju *Apply as Filter*. Možemo i da kombinujemo izraz za *display* filtar, na osnovu vrednosti više polja HTTP paketa, tada koristimo opciju *Prepare a Filter*, tako da imamo mogućnost da proširimo izraz a da se filtriranje ne pokrene automatski kao kod opcije *Apply as Filter*.

📕 P	roba 07092016.pcap							d 🔀
File	Edit View Go Capture	Analyze Statisti	cs Telephony Wireless Tools	Help				
1	🔳 🔬 🕑 🔰 🛅 🔀 🔁 🛛	۹ 👄 📾 🦉	F 👲 🚍 🚍 @ Q Q Q 💷					
hi	ttp						Expression	+ TOP
No.	Time	Source	Source port Destination	Destination port Length	Protocol	Info	Expert	
	126 10.541708	10.0.33.7	54324 ns2.unstops	. 80	176 HTTP	GET /wpad.dat?29f56ccde546d7086198772ce0422c179750276	1 Chat	
	127 10.572671	ns2.unstops	80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK (application/x-ns-proxy-autoconfig)	Chat	
	131 10.589388	10.0.33.7	54324 ns2.unstops	. 80	176 HTTP	GET /wpad.dat?29f56ccde546d7086198772ce0422c17975027) Chat	
	132 10.620344	ns2.unstops	80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK (application/x-ns-proxy-autoconfig)	Chat	
	138 10.624225	10.0.33.7	54332 a1363.dscg.a.	. 80	313 HTTP	GET /pki/crl/products/WinPCA.crl HTTP/1.1	Chat	-
	140 10.625098	a1363.dscg.a.	80 10.0.33.7	54332	280 HTTP	HTTP/1.1 304 Not Modified	Chat	
	141 10.636554	10.0.33.7	54324 ns2.unstops	. 80	176 HTTP	GET /wpad.dat?29f56ccde546d7086198772ce0422c179750276) Chat	
	142 10.668901	ns2.unstops	80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK (application/x-ns-proxy-autoconfig)	Chat	
	144 10.672034	10.0.33.37	61324 239.255.255.	. 1900	216 SSDP	M-SEARCH * HTTP/1.1	Chat	
	145 10.695315	10.0.33.7	54324 ns2.unstops	. 80	176 HTTP	GET /wpad.dat?29f56ccde546d7086198772ce0422c17975027	Chat	
	146 10.726308	ns2.unstops	80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK (application/x-ns-proxy-autoconfig)	Chat	
	147 10.727679	10.0.33.7	54332 a1363.dscg.a.	. 80	331 HTTP	GET /pki/crl/products/MicCodSigPCA_08-31-2010.crl HT	P. Chat	+
	Length: 182							
1 2	Checksum: 0x1c23 [vali	dation disabled	17					
	[Good Checksum: Fal	se]	-					
	[Bad Checksum: Fals	e]						
	[Stream index: 1230]							
4 H	ypertext Transfer Protoc	ol						
1	M-SEARCH * HTTP/1.1\r\	п						
	HOST: 239.255.255.250:	1900\r\n						
	MAN: "ssdp:discover"\r	\n						
	MX: 1\r\n							=
	ST: urn:dial-multiscre	en-org:service:	dial:1\r\n					
	USER-AGENT: Google Chr	ome/52.0.2743.1	16 Windows\r\n					
	\r\n							
	Full request URI: htt	p://239.255.255	.250:1900*					
	[HTTP request 1/4]	42444723						-
0000	INEXC reduest in mane	: 10005/1		-				
0000	0 01 00 50 /T TT TA /4	29 at 00 ce 03	21 75 of ff V D					<u> </u>
0020	ff fa c4 cf 07 6c 00	b6 1c 23 4d 2d	53 45 41 521	-SEAR				Ξ.
0030	43 48 20 2a 20 48 54	54 50 2f 31 2e	31 0d 0a 48 CH * HTT P/1	.1H				
0040	4f 53 54 3a 20 32 33	39 2e 32 35 35	2e 32 35 35 OST: 239 .25	5.255				
0056	2e 32 35 30 3a 31 39	30 30 0d 0a 4d	41 4e 3a 20 .250:190 0	MAN :				-
0	Hypertext Transfer Protocol:	Protocol				Packets: 748043 * Displayed: 2937 (0.4%) * Load ti	ne: 0:11.676 Prof	file: Default
6) 📋 🕘 🛛					SR 🔺 🏦	🕼 😼 ant 🕩 🛛 14.0	0:48 19.2016

Slika 7.8.1. Paketi koji sadrže HTTP zaglavlje

Na slici 7.8.2. prikazan je broj HTTP paketa u jedinici vremena (crvenom bojom) u odnosu na ukupan broj paketa (crnom bojom) u snimljenom fajlu.



Wireshark IO Graphs: Proba 07092016

Slika 7.8.2.Broj paketa koji sadrže HTTP protokol (crvena boja) u odnosu na ukupan broj paketa (crna boja)

Pakete koji sadrže HTTP zaglavlje možemo filtirati u glavnom prozoruna osnovu izraza *http.response.code*==404, a kao rezultat Wireshark će prikazati pakete koji nam govore da tražena stranica na Internetu nije pronađena. Za razliku od tog filtra, možemo da unesemo izraz *http.response.code*==200 i Wireshark će nam izlistati sve pakete koji u sebi sadrže ok kao odgovor na HTTP zahtev (slika 7.8.3.).

📕 Pr	oba 07092016.pcap								
File	Edit View Go Capture	Analyze Statistics	Telephony Wireless	Tools Help					
<u> </u>	I 🔬 🕢 👢 💿 🖄 🖻	९ 🗢 🕾 🕾 👖		Q. 11					
ht	tp.response.code==200							🔀 🔜 💌 Expression	т + тср
No.	Time	Source	Source port	Destination	Destination port Length	Protocol	Info	Expert	*
	127 10.572671	198.16.79.61		80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK	(application/x-ns-proxy-aut Chat	
	132 10.620344	198.16.79.61		80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK	(application/x-ns-proxy-aut Chat	
	142 10.668901	198.16.79.61		80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK	(application/x-ns-proxy-aut Chat	
	146 10.726308	198.16.79.61		80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK	(application/x-ns-proxy-aut Chat	
	150 10.793746	198.16.79.61		80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK	(application/x-ns-proxy-aut Chat	
	153 10.851249	198.16.79.61		80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK	(application/x-ns-proxy-aut Chat	
	157 10.926921	198.16.79.61		80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK	(application/x-ns-proxy-aut Chat	
	159 10.989135	198.16.79.61		80 10.0.33.7	54324	379 HTTP	HTTP/1.1 200 OK	(application/x-ns-proxy-aut. Chat	
	1262 32.435386	50.7.182.197	56	408 10.0.33.7	54345	1414 HTTP	HTTP/1.1 200 OK	Chat	
	1309 32.945365	65.52.144.16		80 10.0.33.7	54349	1327 HTTP	HTTP/1.1 200 OK	(text/html) Chat	
	1342 33.659757	65.52.144.16		80 10.0.33.7	54349	1488 HTTP	HTTP/1.1 200 OK	(image/x-icon) Chat	
	1348 33.770007	213.199.133.147		80 10.0.33.7	54352	875 HTTP	HTTP/1.1 200 OK	Chat	
	4354 33 850003	313 100 133 147		00 10 0 33 7	54353	MAC UTTO	UTTO (1 4 300 OV	(incenter incent) (these	1-
P Fr	ame 1262: 1414 bytes o	n wire (11312 bits)	, 1414 bytes captu	red (11312 bits)				-
PET	mernet II, Src: 10.0.3	2.1 (90:1/:ac:bc:6c	:cd), Dst: 10.0.3	./ (78:24:af:9a	:00:88)				
P In	iternet Protocol Version	1 4, Src: 50.7.182.	197, Dst: 10.0.33.	./					
PIL	ansmission Control Pro-	tocol, Src Port: 56	408 (56408), Dst H	ort: 54345 (543	45), Seq: 1, Ack: 456	, Len: 1360			
A HÀ	pertext Transfer Proto	co1							
P	HTTP/1.1 200 OK\r\n								
	Server: openresty(r(n								=
	Date: Wed, 0/ Sep 2010	5 14:57:15 GMI \r\n							
	Transfer-Encoding: chi	inked\r\n							
	Connection: close \r \n								
	Ar An								
	[HITP response 1/1]								
	[lime since request: 4	3.205010000 seconds]						
	[Request in Trame: 12:	58							
P	HITP chunked response								
	Data (1226 Dytes)			-					
0000	78 24 af 9a d0 88 90	17 ac bc 6c cd 08	00 45 08 x\$	E.					*
0010	05 /8 a8 /5 40 00 32	06 8/ 27 32 0/ 06	C5 0a 00 .x.ug	Z/2					
0020	00 77 5f b2 00 00 48	54 54 50 2f 31 2e	31 20 32 W	HT TP/1 1 2					
0040	30 30 20 4f 4b 0d 0a	53 65 72 76 65 72	3a 20 6f 00 0K	.S erver: o					*
Fram	e (1414 bytes) De-chunked e	entity body (1226 bytes)							
•	Proba 07092016						Packets: 7480	043 • Displayed: 464 (0.1%) • Load time: 3:13.170	Profile: Default
									22:03
								🔺 🤷 👘 👘	14.09.2016

Slika 7.8.3.Paketi koji sadrže HTTP zaglavlja sa odgovorom 200 - OK

Ako u polje *display* filtra unesemo sledeći izraz *http.request.method*=="*GET*", kao rezultat dobićemo pakete koji u sebi sadrže metodu GET. Ako želimo da vidimo pakete koji sadrže metodu POST, možemo koristiti izraz *http.request.method*=="*POST*"za filtriranje.

Izuzetno koristan filtar u Wireshark aplikaciji je *http.time* koji nam pokazuje vreme odziva HTTP protokola.Ako u I/O Graph polju za filtriranje unesemo *http.time* i funkcija AVG(*), dobićemo grafik koji prikazuje prosečno vreme odziva HTTP protokola (slika 7.8.4.).

Wireshark IO Graphs: Proba 07092016



Ako u polju za *display* filtar unesemo izraz *http.time>1*, prikazaće nam se paketi kod kojih je vreme odziva veće od jedne sekunde (što bi moglo da predstavlja problem), slika 7.8.5.

	Deales 07000016 a sea										
-	Proba 07092010.pcap										
File	e Edit View Go Capture	Analyze Statistics	Telephony Wireless	lools Help							
1	🔳 🖉 🛞 📙 🛅 🔀 🙆	९ 🗢 🗢 🖀 🚹 🦉	l 🗖 🗐 🔍 Q (Q, 111							
	http.time>1									Expression	п + тср
No.	Time	Source	Source port	Destination	Destination port	Length	Protocol	Info		Expert	
	18305 308.974056	198.16.79.61		80 10.0.33.7	54526		379 HTTP	HTTP/1.1 200 OK	(application/x-	ns-proxy-aut Chat	
	25087 361.784083	194.48.48.70		80 10.0.33.7	54550		1380 HTTP	HTTP/1.1 200 OK	(PNG)	Chat	
	58243 432.013928	5.63.150.180		80 10.0.33.7	54631		752 HTTP	HTTP/1.1 200 OK	(text/html)	Chat	
	136787 605.483463	5.63.150.180		80 10.0.33.7	54729		989 HTTP	HTTP/1.1 200 OK	(text/html)	Chat	
	733869 882.001541	173.194.151.204		80 10.0.33.7	54855		77 MP4			Chat	
	Server: openresty/1.9. Date: Wed, 07 Sep 2010 Content-Type: text/pla Doment-Length: 54\r\r Connection: keep-alive	7.4\r\n 5 15:01:47 GMT\r\n din\r\n 6 6									
	Expires: Wed, 07 Sep 2 Cache-Control: max-age Content-Type: applicat \r\n	016 17:01:47 GMT\r =7200\r\n :ion/x-ns-proxy-aut	\n oconfig\r\n								
	[HTTP response 1/8] [Time since request: 1 [Request in frame: 183	558935000 seconds	1								E
	[Next request in frame [Next response in frame	: 18310] :: 18324]									
P	Line-based text data: app	ication/x-ns-prox	y-autoconfig								-
00 00 00 00 00	100 78 24 af 9a de 88 90 10 01 6d a2 b7 40 00 32 120 21 07 00 50 d4 f8 fa 130 00 c5 57 de 00 04 48 140 30 30 20 4f 4b 0 0a 140 30 30 20 4f 4b 0d 0a	17 ac bc 6c cd 08 06 64 77 c6 10 4f a0 31 a9 b9 30 db 54 54 50 2f 31 2e 53 65 72 76 65 72 29 3f 31 20 20	00 45 08 x\$ 3d 0a 00 .m@. f1 50 18 !P 31 20 32	E. .2. dw0= 							A H
00	10 05 00 72 05 75 74		or action pennes					11 .			*
_	Proba 07092016	1 V						Packets: 748	043 • Displayed: 5 (0.0%	6) * Load time: 0:11.534	Profile: Default
6	🦻 📜 🕘 🛛	<u>i</u>	W							EN 🔺 🛕 🕼 😼 all 🕩	22:31 14.09.2016

Slika 7.8.5.HTTP paketi koji imaju vreme odziva veće od jedne sekunde

Ako želimo, možemo desnim klikom na polje [Time since request] da izaberemo opciju *Apply as Column* kako bi se vreme odziva HTTP protokola (*http.time*) prikazalo kao posebna kolona u glavnom prozoru.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst star
Intel HTTP Packets	2851				0.0031	100%	0.2700	361.612
Other HTTP Packets	18				0.0000	0.63%	0.0700	360.662
HTTP Response Packets	545				0.0006	19.12%	0.1500	355.895
???: broken	0				0.0000	0.00%	-	-
4 5xx: Server Error	2				0.0000	0.37%	0.0200	356.614
500 Internal Server Error	2				0.0000	100.00%	0.0200	356.614
4 4xx: Client Error	14				0.0000	2.57%	0.0300	490.780
408 Request Time-out	7				0.0000	50.00%	0.0300	490.780
404 Not Found	6				0.0000	42.86%	0.0200	305.350
400 Bad Request	1				0.0000	7.14%	0.0100	245.072
▲ 3xx: Redirection	57				0.0001	10.46%	0.0400	198.170
304 Not Modified	29				0.0000	50.88%	0.0400	198.170
302 Found	26				0.0000	45.61%	0.0400	201.281
301 Moved Permanently	2				0.0000	3.51%	0.0100	307.232
A 2xx: Success	471				0.0005	86.42%	0.1500	355.895
204 No Content	7				0.0000	1.49%	0.0100	198.180
200 OK	464				0.0005	98.51%	0.1500	355.895
4 1xx: Informational	1				0.0000	0.18%	0.0100	403.739
101 Switching Protocols	1				0.0000	100.00%	0.0100	403.739
HTTP Request Packets	2288				0.0025	80.25%	0.1500	355.813
SEARCH	1329				0.0014	58.09%	0.0600	390.802
POST	12				0.0000	0.52%	0.0200	406.121
OPTIONS	3				0.0000	0.13%	0.0100	198.168
NOTIFY	400				0.0004	17.48%	0.0300	283.621
GET	540				0.0006	23.60%	0.1500	355.813
CONNECT	4				0.0000	0.17%	0.0100	403.157

Slika 7.8.6.Ukupan broj HTTP paketa, kao i broj HTTP paketa sa različitim kodovima

Ako kliknemo na opciju Statistics->HTTP->Packet Counter u meniju, otvoriće se prozor prikazan na slici 7.8.6.

Rezultat koji nam Wireshark prikazuje možemo da sačuvamo kao *.csv fajl klikom na Save As... dugme. Ako otvorimo *.csv fajl u programu Excel, možemo da prikažemo broj HTTP paketa grafički (slika 7.8.7.).



Slika 7.8.7. Pregled broja HTTP paketa

Zanimljivo je poređenje broja paketa koji sadrže HTTP zaglavlje u odnosu na broj paketa koji sadrže HTTPS (*HTTP Secure, SSL*) zaglavlje u jedinici vremena. U snimljenom fajlu se primećuje da imamo znatno više paketa koji se prenose na siguran način kroz mrežu, uz odgovarajuće šifrovanje poverljivih informacija (slika 7.8.8.).

Wireshark IO Graphs: Proba 07092016



Slika 7.8.8.Poređenje broja HTTP paketa (crvena linija) i SSL paketa (plava linija) u jedinici vremena

Na slici 7.8.9 prikazan je broj paketa koji sadrže HTTP zaglavlje u odnosu na broj paketa koji sadrže TCP zaglavlje u jedinici vremena.



Wireshark IO Graphs: Proba 07092016



8.ZAKLJUČAK

Analiziranje paketa pomaže u razumevanju mrežnih karakteristika, proveri ko je sve na mreži, utvrđivanje ko ili šta koristi dostupni protok, identifikovanju pikova kada se mreža najviše koristi, identifikovanju mogućih napada ili zlonamernih aktivnosti i pronalaženju nesigurnih aplikacija.

Wireshark je jedan od najboljih besplatnih programskih alata za analizu mrežnog saobraćaja. Podržava sve važnije mrežne protokole i ima mogućnost nadogradnje za nove protokole.Kako je Wireshark open source model, podrška za nove protokole se dodaje prilikom svake naprednije i novije verzije aplikacije. Snimljeni saobraćaj može se naknadno analizirati, iz sačuvanog .pcap fajla.Wireshark ima grafički korisnički interfejs sa veoma jasnim menijem i opcijama koje nudi. Pored toga, pruža i nekoliko funkcija koje poboljšavaju korišćenje same aplikacije, kao što su razdvajanje protokola na osnovu boja i detaljnu grafičku prezentaciju prikupljenih paketa. Wireshark podržava većinu modernih operativnih sistema, uključujući Windows, Mac OS X i Linux platforme.

Wireshark nudi mogućnost filtriranja tokom samog snimanja paketa (*capture* filtri), kao i nakon što se samo snimanje završi (*display* filtri). Filtriranjem možemo suziti mrežni saobraćaj samo na one pakete koji nas interesuju i time olakšati samu analizu. Možemo filtrirati na osnovu različitih kriterijuma, kao što su MAC i IP adresa, port koji koriste protokoli za slanje i primanje paketa, sadržaj samih paketa, tip paketa, veličina paketa, protokoli čije se zaglavlje može naći u paketu.

Najlakši način da se vrši analiza snimljenog saobraćaja jeste korišćenjem grafičkih rešenja koje nudi Wireshark ili eksportovanjem različitih rezultata u format pogodan za dalju analizu.

Snimani su paketi na personalnom računaru u poslovnom okruženju. Navedeni filtri, analize i skripte u ovom radu mogu da se implementiraju kako u jednostavnim, tako i u složenijim mrežama. Prilikom analiziranja paketa, bitno je naći sve sumnjive pakete i korisnike, utvrditi da li ima mrežnih zagušenja i zbog čega nastaju kako bismo ih uspešno otklonili.

Wireshark dolazi uz ugrađeni Lua programski jezik. Lua je moćan, jednostavan i prenosiv skriptni jezik koji se može koristiti za nadogradnju Wireshark funkcija. Ako neki protokol nije definisan u okviru samog Wireshark programa, postoji mogućnost da se definiše kroz Lua skriptu i implementira kako bi Wireshark mogao da ga ispravno interpretira.Razlikujemo Lua *Listener Taps* koji se koriste za statistiku i analizui Lua *Dissectors* koji se koriste za nove protokole. Primeri Lua programskog koda dati su kao prilog uz ovaj rad.

Pre samog snimanja paketa, potrebno je proveriti da li imamo dozvolu za to. U rukama zlonamernih korisnika Wireshark program se može koristiti i za narušavanje bezbednosti i sigurnosti same mreže. Kako snimamo na nivou paketa, dostupan nam je sadržaj svih paketa koji se razmene. Ako postoje neki podaci kao što su šifre i korisnička imena koja nisu enkriptovana, može doći do zloupotreba pomenutih podataka.

Wireshark se ne može koristiti preventivno, u smislu sprečavanja problema do kojih može doći u mreži, već samo za naknadnu analizu, da li u realnom vremenu ili kasnije. Tek nakon analiziranja možemo primetiti neke nepravilnosti i probleme do kojih može doći.

Iako ima mane, zbog svega navedenog Wireshark predstavlja najbolji izbor za snimanje i analizu mrežnog saobraćaja.

LITERATURA

- [1] C. Sanders, Practical Packet Analysis, 2nd Edition, No Starch Press, 2011.
- [2] L. Chappell, *Wireshark Network Analysis,2nd Edition*, Protocol Analysis Institute Inc, dba Chappell University, 2012.
- [3] M. Stojanović, V. Aćimović-Raspopović, *Savremene IP mreže: Arhitekture, Tehnologije i Protokoli*, Akademska Misao, 2012.
- [4] R. Ierusalimschy, Programming in Lua, 3rd Edition, Lua.Org, 2013.
- [5] Introduction to Wireshark 2.0 w/ Gerald Combs and Laura Chappell[Online]. Available: https://www.youtube.com/watch?v=rLfYuO6pdVA
- [6] *SharkFest'15 Hadriel Kaplan Class 11*[Online]. Available: <u>https://www.youtube.com/watch?v=HTtVHxIh6ww</u>
- [7] Wireshark oficijelna stranica[Online]. Available: https://www.wireshark.org/
- [8] Wireshark Wiki [Online]. Available: https://wiki.wireshark.org/
- [9] Wireshark Wiki Lua[Online]. Available: https://wiki.wireshark.org/Lua

A. Prilozi

Lua predstavlja jednostavan, ali moćan skriptni programski jezik. Lua programski jezik se koristi u različite svrhe: za pisanje igrica (kao što je npr. popularna *World of Warcraft* - WoW), za pisanje proširenja za programe, za pisanje testova, za pisanje konfiguracija i opisivanje podataka. Kreiran je na *Pontifical Catholic* univerzitetu, u Brazilu.

Lua je jednostavan programski jezik zbog čega je pogodan i za početnike. Pored toga, Lua je kompaktan programski jezik koji poseduje jako malu standardnu biblioteku, a pogodan je za upotrebu na namenskim uređajima. Programski jezik Lua poseduje mehanizme za širok skup funkcionalnosti i lako se spreže sa drugim programskim jezicima.

Lua programski jezik odlikuje dinamičko tipiziranje, automatsko upravljanje memorijom, funkcije koje se tretiraju kao tipovi, mehanizimi za konkurentno izvršavanje programa, kao i mehanizmi za objektno, funkcionalno i proceduralno programiranje. Promenljive mogu biti lokalne i globalne, koriste se iteratori, moduli (biblioteke), tabele i metatabele. Tabele su osnovna struktura podataka u Lui. Ostale strukture, kao i objekti/klase, se predstavljaju tabelama.

Članove Lua zajednice možete pronaći na sledećem linku <u>www.lua-users.org</u>, odgovarajuću mailing listu na linku <u>http://www.lua.org/lua-l.html</u>, a Lua priručnik na sledećoj veb adresi <u>http://www.lua.org/manual/5.3/</u>.

Lua (zvanični kompajler) i LuaJIT (*Just In Time*) su najpoznatiji kompajleri za Luu. Postoji nekoliko verzija programskog jezika Lua – 5.1, 5.2 i 5.3 su najčešće implementirane.Programi pisani za različite verzije Lue su nekada nekompatibilni. Interpreter prevodi kod u toku izvršavanja i direktno izvršava operacije definisane u izvornom programu nad ulaznim podacima. Prednosti interpretera su u tome što koristi mnogo manje memorije i omogućava bolju dijagnostiku grešaka i interaktivno ispitivanje.Mane interpretera su zahtevanje više vremena za izvršavanje ulaznog programa i svako novo izvršavanje ponavlja kompletno prevođenje.

Identifikatori u Lua programskom jeziku mogu počinjati malim slovom, velikim slovom ili znakom_, a mogu sadržati i cifre pored navedenih znakova. Ime bi trebalo da oslikava upotrebu promenljive.Treba izbegavati identifikatore koji počinju sa donjom crtom i jednim ili više velikih slova iza nje, zato što su oni rezervisani.

Sledeće reči su rezervisane i ne mogu da se koriste kao identifikatori: *and, end, if, or, until, break, false, in, repeat, while, do, goto, local, return, else, for, nil, then, elseif, function, not, true.* Lua programski jezik pravi razliku između malih i velikih slova.

Komentare možemo postaviti bilo gde sa (--) koji važe sve do kraja linije. Lua dozvoljava i komentare u bloku koji počinju sa --[[i važe sve do sledećeg]].

Globalne promenljive ne moramo da definišemo, jednostavno ih koristimo bez definisanja.Nije greška ako pristupamo promenljivoj koja nije definisana, samo ćemo dobiti da joj je vrednost *nil*.Pored globalnih promenljivih, Lua podržava i lokalne promenljive, koje se kreiraju uz ključnu reč *local*. Za razliku od globalnih promenljivih, lokalne promenljive važe samo u bloku u kojem su definisane.

Postoji osam osnovnih tipova u Lua programskom jeziku: *nil, boolean, number, string, userdata, function, thread* i *table*. Promenljive nemaju predefinisane tipove, bilo koja promenljiva može sadržati vrednosti bilo kog tipa.

Tabele čineosnovnu strukturu podataka u Lua programskom jeziku. Tabele možemo da posmatramo kao dinamički alociran objekat, a program koristi pokazivače na te objekte. Konstruktori su izrazi koji kreiraju tabele u Lui. Najjednostavniji konstruktor je {}, koji kreira praznu tabelu. Kontruktori mogu da inicijalizuju listu koja se unosi u tabelu. Tabele koristimo kako bi predstavili nizove, zapise i ostale strukture podataka na jednostavan, uniforman i efikasan način.

Funkcije predstavljaju prvoklasne promenljive – programi mogu da čuvaju funkcije u okviru promenljivih, da koriste funkciju kao argument neke druge funkcije ili da vrate funkciju kao rezultat. Lua može da poziva funkcije koje su napisane u Lua programskom jeziku, kao i funckije napisane u C programskom jeziku. Sve standardne biblioteke u Lui su napisane u C programskom jeziku.

Lua podržava standardne aritmetičke operatore, kao što su + (plus), - (minus), * (množenje), / (deljenje), ^(eksponent), % (procenat) i _ (negacija). Svi aritmetički operatori rade sa realnim brojevima. Lua podržava sledeće relacione operatore: <, >, <=, >=, == i ~=. Svi relacioni operatori kao rezultat daju *boolean* vrednost (*true* ili *false*). Ako vrednosti imaju različite tipove, Lua smatra da nisu jednaki. U suprotnom, Lua ih poredi po tipovima. Lua podržava i logičke operatore *and*, *or* i *not*. Svi logički operatori posmatraju tipove *boolean* i *nil* kao *false*, a sve ostalo kao *true*. Operator *and* kao vrednost vraća prvi argument ako je on *false*, u suprotnom vraća drugi argument. Operator *or* vraća prvi argument ako nije *false*, u suprotnom vraća drugi argument. Operator *not* uvek vraća *boolean* vrednost.

Lua podržava konvencionalni set izraza, sličnih onim u programskom jeziku C. Konvencionalni izrazi uključuju dodelu vrednosti, kontrolne strukture i proceduralne pozive. Dodela vrednosti predstavlja osnovni vid promene vrenosti promenljive ili polja tabele. Kontrolne strukture mogu biti tipa *if* za uslovno izvršavanje i *while*, *repeat* i *for* za *for* iteracije. Sve kontrolne strukture imaju eksplicitni terminator – *end* označava kraj *if*, *for* i *while* struktura, dok *until* označava kraj *repeat* struktura.

Lua programski jezik je ugrađen u samu Wireshark aplikaciju. Fajl init.lua u folderima*global configuration* i *personal configuration* se pokreće prilikom svakog otvaranja Wireshark programskog alata. Tom prilikom pokreću se i sve lua skripte sa *.lua ekstenzijom. Skriptu možemo pokrenuti iz komandne linije sledećom naredbom –*X lua_script:xxx.lua* (gde xxx predstavlja ime *.lua fajla u kojem je sačuvana skripta koju želimo da pozovemo).

Ako kliknemo na opciju About u meniju pojaviće nam se prozor kao na slici A.1.1. sa osnovnim informacijama o Wireshark programskom alatu.

Lua programski jezik se koristi u okviru Wireshark programa kao *dissector* – koristi se za dekodiranje paketa podataka, *post-dissector* – poziva se nakon što se pozovu svi ostali *dissector*-i, *listener* – koristi se za prikupljanje informacija nakon što je paket seciran.*Listener* u okviru Lua skripte koristimo za prikupljanje statistike i analizu, dok se za definisanje novih protokola koristi *dissector*.

Po potrebi, u init.lua fajlu možemo da nađemo sledeći red *disable_lua = false* i da setujemo vrednost na *true*, ako ne želimo da Wireshark pokreće Lua skripte.



Slika A.1.1. Na kartici About Wireshark možemo da vidimo koja verzija Lua skripte je integrisana u sam Wireshark (u našem slučaju Lua 5.2)

A.1. Lua skripta za registrovanje portova 4889-4893 za HTTP protokol

Wireshark ima unapred definisane karakteristike svih protokola koje podržava. Na osnovu tih karakteristika Wireshark prepoznaje protokole i obeležava pakete prilikom snimanja mrežnog saobraćaja.

Ako prilikom analiziranja snimljenog saobraćaja naiđemo na pakete koji imaju *source* ili *destination* port koji Wireshark ne prepoznaje kao port koji koristi HTTP protokol, možemo da napišemo Lua programski kod kojim te portove registrujemo kao portove koje koristi HTTP protokol. Nakon pokretanja koda, Wireshark prepoznaje sve pakete koji sadrže definisane portove kao pakete HTTP protokola, što može znatno da nam olakša sam prikaz snimljenog saobraćaja, kao i analizu snimljenih paketa.

Potrebno je da definišemo lokalnu promenljivu *tcp_port_table* u vidu tabele u koju se upisuju svi portovi koje koristi TCP protokol. Nakon toga, definišemo lokalnu promenljivu *http_dissector* koja nam govori da HTTP protokol koristi port 80 iz tabele sa TCP portovima za

prenos paketa. Napravimo petlju koja za određene portove (4890, 4891, 4892 i 4893) izvršava dodavanje datih portova u lokalnu promenljivu*http_dissector*, čime definišemo da se radi o HTTP protokolu koji se prenosi preko definisanih portova.

Možemo da koristimo različite portove za različite protokolepo istom principu, u zavisnosti od potrebe.

```
do
local tcp_port_table = DissectorTable.get("tcp.port")
local http_dissector = tcp_port_table:get_dissector(80)
for i,port in ipairs{4890,4891,4892,4893} do
        tcp_port_table:add(port,http_dissector)
end
end
```

A.2. Lua skripta za brojanje paketa koji dolaze od/do IP adrese 10.0.33.7

do

```
packets = 0;
local function init_listener()
local tap = Listener.new("frame","ip.addr == 10.0.33.7")
function tap.reset()
packets = 0;
end
function tap.packet(pinfo,tvb,ip)
packets = packets + 1
end
function tap.draw()
print("Packets to/from 10.0.33.7",packets)
end
end
init_listener()
end
```

A.3. Lua skripta za definisanje protokola DNS na portu 65333

local dns = Proto("mydns", "MyDNS Protocol")

local pf_trasaction_id = ProtoField.new ("Transaction ID", "mydns.trans_id", ftypes.UINT16)

local pf_flags = ProtoField.new ("Flags", "mydns.flags", ftypes.UINT16, nil, base.HEX)

local pf_num_questions= ProtoField.uint16("mydns.num_questions", "Number of Questions")

local pf_num_answers = ProtoField.uint16("mydns.num_answers", "Number of Answer
RRs")

local pf_num_authority_rr= ProtoField.uint16("mydns.num_authority_rr", "Number of Authority RRs")

local pf_num_additional_rr = ProtoField.uint16("mydns.num_additional_rr", "Number of
Additional RRs")

```
dns.fields = {
  pf_trasaction_id;
  pf_flags;
  pf_num_questions;
  pf_num_answers;
  pf_num_authority_rr;
  pf_num_additional_rr;
 }
```

```
function dns.dissector(tvbuf, pktinfo, root)
pkinfo.cols.protocol:set("MyDNS")
local pktlen = tvbuf:reported_length_remaining()
local tree = root:add(dns, tvbuf:range(0,pktlen))
tree:add(pf_trasaction_id, tvbuf:range(0,2))
```

```
local transid = tvbuf:range(0,2):uint()
pktinfo.cols.info:set("("..transid..")")
```

```
tree:add(pf_flags, tvbuf:range(2,2))
```

tree:add(pf_num_questions, tvbuf:range(4,2))
tree:add(pf_num_answers, tvbuf:range(6,2))
tree:add(pf_num_authority_rr, tvbuf(8,2))
tree:add(pf_num_additional_rr, tvbuf:range(10,2):range()())

return pktlen end

DissectorTable.get("udp.port"):add(65333, dns)

A.4. Lua skripta za statističku analizu protokola

do

ip_addr_extractor = Field.new("ip.addr")
tcp_src_port_extractor = Field.new("tcp.srcport")
tcp_dst_port_extractor = Field.new("tcp.dstport")
udp_port_extractor = Field.new("udp.port")
icmp_type_extractor = Field.new("icmp.type")
icmp_code_extractor = Field.new("icmp.code")

local function init_listener()
local tap = Listener.new("frame")

-- tcp port counts

local ipv4_tcp_src_cache = {}
local ipv4_tcp_dst_cache = {}
local ipv4_tcp_src_count = 0
local ipv4_tcp_dst_count = 0
function stats_ipv4_tcp_port_counts()
local tcp_src_port
local tcp_dst_port
tcp_src_port = tcp_src_port_extractor()
tcp_dst_port = tcp_dst_port_extractor()

```
if (tcp src port) then
  if (not ipv4 tcp src cache[ tostring( tcp src port ) ] == true ) then
    ipv4_tcp_src_cache[ tostring( tcp_src_port ) ] = true
    ipv4 tcp src count = ipv4 tcp src count + 1
  else
    -- print("tcp src port already recorded")
  end
else
  -- print("no tcp src port")
end
if (tcp dst port) then
  if (not ipv4 tcp dst cache[ tostring( tcp dst port ) ] == true ) then
    ipv4_tcp_dst_cache[ tostring( tcp_dst_port ) ] = true
    ipv4_tcp_dst_count = ipv4_tcp_dst_count + 1
    -- print("tcp dst port new: ".. tostring(tcp dst port))
  else
    -- print("tcp dst port old: ".. tostring(tcp dst port))
  end
else
  -- print("tcp_dst_port none: ".. tostring(tcp_dst_port))
end
```

```
end
```

-- udp port counts

local ipv4_udp_src_cache = {}
local ipv4_udp_dst_cache = {}
local ipv4_udp_src_count = 0
local ipv4_udp_dst_count = 0
function stats_ipv4_udp_port_counts()
 local udp_src_port
 local udp_dst_port
 udp_src_port, udp_dst_port = udp_port_extractor()
 if (udp_src_port) then

```
if (not ipv4 udp src cache[ tostring( udp src port ) ] == true ) then
       ipv4 udp src cache[tostring(udp src port)] = true
       ipv4 udp src count = ipv4 udp src count + 1
    else
       -- print("udp src port already recorded")
    end
  else
    -- print("no udp src port")
  end
  if (udp dst port) then
    if (not ipv4 udp dst cache[ tostring( udp dst port )] == true ) then
       ipv4 udp dst cache[ tostring( udp dst port ) ] = true
       ipv4_udp_dst_count = ipv4_udp_dst_count + 1
       -- print("udp dst port new: ".. tostring(udp dst port))
    else
       -- print("udp dst port old: ".. tostring(udp dst port))
    end
  else
    -- print("udp dst port none: ".. tostring(udp dst port))
  end
end
```

-- icmp type code counts

local ipv4_icmp_type_cache = {}
local ipv4_icmp_type_count = 0
function stats_icmp_type_counts(pinfo,tvb)
local icmp_type
local icmp_code
icmp_type = icmp_type_extractor()
icmp_code = icmp_code_extractor()
if(icmp_type and icmp_code) then
 if(not ipv4_icmp_type_cache[tostring(icmp_type) ... '-' ... tostring(icmp_code)]
there

== true) then

```
ipv4_icmp_type_cache[ tostring( icmp_type ) ... '-' .. tostring( icmp_code ) ] =
    ipv4_icmp_type_count = ipv4_icmp_type_count + 1
    else
        -- print("icmp type and code already recorded")
    end
    else
        -- print("no icmp type and code")
    end
end
```

-- ipv4 counts

```
local ipv4 src cache = \{\}
local ipv4 dst cache = \{\}
local ipv4 src count = 0
local ipv4 dst count = 0
function stats ipv4 counts(pinfo,tvb)
  local ip src
  local ip dst
  ip src, ip dst = ip addr extractor()
  if ( ip_src ) then
     if (not ipv4_src_cache[ tostring(ip_src) ] == true ) then
       ipv4 src cache[ tostring(ip src) ] = true
       ipv4 src count = ipv4 src count + 1
     else
       -- print("src already recorded")
     end
     --- try counting tcp/udp and icmp once for every ipv4 pkt
     if (pinfo.ipproto == 1) then
       stats icmp type counts(pinfo,tvb)
     elseif(pinfo.ipproto == 6) then
       stats_ipv4_tcp_port_counts()
     elseif (pinfo.ipproto == 17) then
       stats ipv4 udp port counts()
```

true

```
end
else
-- print("NO src")
end
if ( ip_dst ) then
    if ( not ipv4_dst_cache[ tostring(ip_dst) ] == true ) then
        ipv4_dst_cache[ tostring(ip_dst) ] = true
        ipv4_dst_count = ipv4_dst_count + 1
        else
        -- print("dst already recorded")
        end
else
        -- print("NO dst")
end
end
```

```
-- start/end times
```

```
local start_time
local end_time
function stats_start_end_times(pinfo)
    if (not start_time) then
        start_time = pinfo.abs_ts
        end_time = pinfo.abs_ts
    else
        if (start_time > pinfo.abs_ts) then start_time = pinfo.abs_ts end
        if (end_time < pinfo.abs_ts) then end_time = pinfo.abs_ts end
    end
end</pre>
```

function tap.reset() end

```
function tap.packet(pinfo,tvb,ip)
stats_ipv4_counts(pinfo,tvb)
```

```
stats_start_end_times(pinfo)
end
```

========") end

end

init_listener()
end