Univerzitet u Nišu  
Elektronski fakultet  
Katedra za računarstvo

Aleksandar Stamenković

**Forenzika mrežnog saobraćaja**

**Seminarski rad**

**Master akademske studije**

**Predmet: Digitalna forenzika**

Smer: Računarstvo i informatika  
Modul: Bezbednost računarskih sistema

Student:

Aleksandar Stamenković  
Br. indeksa: 1403

Niš, mart 2022. god.

**Sadržaj**

[1 Uvod 3](#_Toc99491925)

[2 Pojam mrežne forenzike 4](#_Toc99491926)

[3 Izazovi mrežne forenzike 4](#_Toc99491927)

[3.1 Karakteristike mrežnog saobraćaja 4](#_Toc99491928)

[3.1.1 IP protokol 4](#_Toc99491929)

[3.1.2 TCP protokol 6](#_Toc99491930)

[3.1.3 UDP protokol 7](#_Toc99491931)

[3.1.4 DNS protokol 8](#_Toc99491932)

[4 Postojeća softverska rešenja 9](#_Toc99491933)

[4.1 TCPDUMP 9](#_Toc99491934)

[4.2 Wireshark 9](#_Toc99491935)

[4.3 NMAP (Zenmap) 10](#_Toc99491936)

[4.4 NetWitness 11](#_Toc99491937)

[5 Literatura 12](#_Toc99491938)

# Uvod

U današnje vreme se računarstvo i digitalizacija podataka nperestano povećava iz dana u dan. Svi ti podaci se u najvećem delu prenose preko računarskih mreža. Nekad se javlja potreba za praćenjem i analiziranjem tih podataka, odnosno njihovog prenosa. Postoje razni alati koji ovako nešto omogućavaju, pa će stoga biti opisani u nastavku ovog rada.

U ovom radu će biti reči o samom pojmu mrežne forenzike, o postojećim alatima kojima je moguće izvršiti tako nešto. Takoće će biti dat razlog nastanka ove tehnike, koja ograničenja se tu javljaju, kao i potrebne funkcionalnosti koje treba da ispune alati koji se koriste u te svrhe.

Nakon toga će biti dat prikaz implementacije alata za mrežnu forenziku koja će biti rađena u Microsoft .NET tehnologiji (verzija 6.0). Takođe će biti dato uputstvo za korišćenje alata, kao i primer korišćenja na konkretnom računaru.

# Pojam mrežne forenzike

Forenzika računarskih mreža u osnovi podrazumeva sistematsko praćenje odlaznog i dolaznog saobraćaja na različitom nivou. Tako, razlikujemo nekoliko vrsta nivoa praćenja mrežnog saobraćaja, a to su:

* Svaki bit – kompletna slika, zahteva ogroman prostor na disku (retko se koristi)
* Podaci o sesijama – prate se različite stvari: mrežne adrese, fizičke (MAC) adrese, zahtevi, portovi, itd.
* Filtriranje – prate se alerti, ključne reči, filtri, itd.
* Statistički nivo – prate se patterni u saobraćaju, filtriranje na nekom od ISO/OSI sloja, empirijski, itd.

# Izazovi mrežne forenzike

## Karakteristike mrežnog saobraćaja

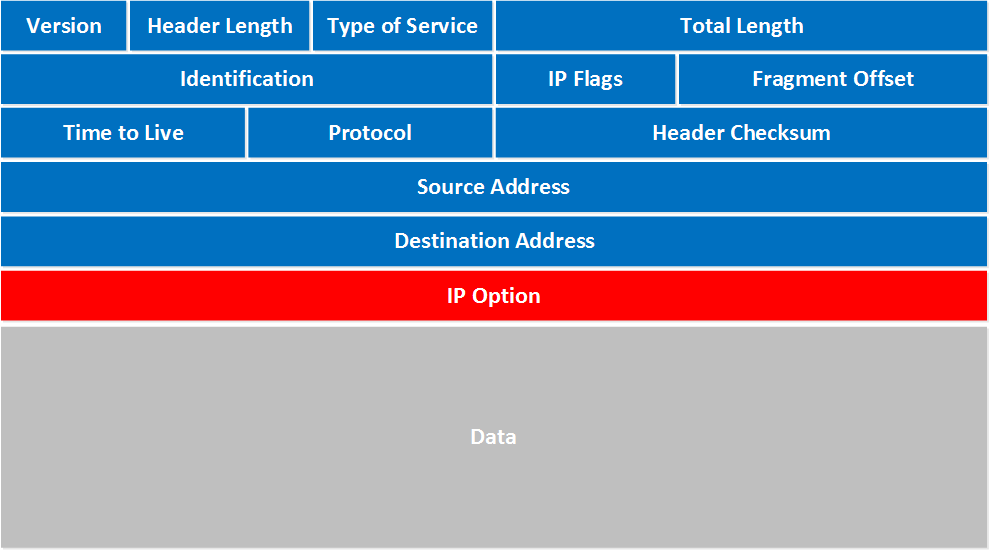
U zavisnosti od protokola koji se koristi, mrežni protokol može biti kriptovan (npr. HTTPS, SSH), ili nekriptovan. Takođe, vrlo često je mrežni saobraćaj fragmentiran. Npr. HTML dokument se ne salje izjedno, već deo po deo ukoliko je sadržaj veći.

Jedna od ključnih stvari u mrežnoj forenzici jeste poznavanje topologije mreže u kojoj su snimljeni podaci. Tako je potrebno znati da li su adrese koje se koriste javne ili privatne, način funkcionisanja adresa (privatnih u javne), DHCP i dinamičko dodeljivanje IP adresa, zatim poznavanje MAC adresa i sl.

Kada je reč o izvorima podataka koji se prikupljaju i analiziraju najčešće se prikupljaju takozvani raw podaci prikupljeni sa mrežnog interfejsa. Naime, svaki od mrežnih protokola ima sistematski definisanu strukturu zaglavlja i tela paketa koji prenosi. S obzirom na to da će se u ovom radu analizirati IP, TCP, UDP i DNS protokoli, u nastavku su data zaglavlja tih protokola.

S obzirom na to da je analiza kompletnog mrežnog saobraćaja gotovo nemoguća zbog njegovog obima, postojeća softverska rešenja nude razne filtere po kojima se mogu selektovati željeni atributi za praćenje mrežnog saobraćaja. Obično je dobar pristup najpre uspostaviti osnovnu sliku performansi strukture saobraćaja na mreži kako bi se identifikovala odstupanja.

### IP protokol

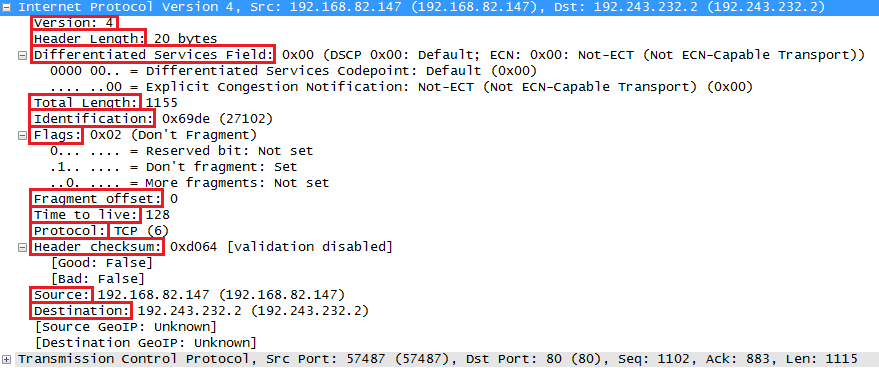


Slika 1 IPv4 zaglavlje

IP zaglavlje se sastoji od sledećih polja:

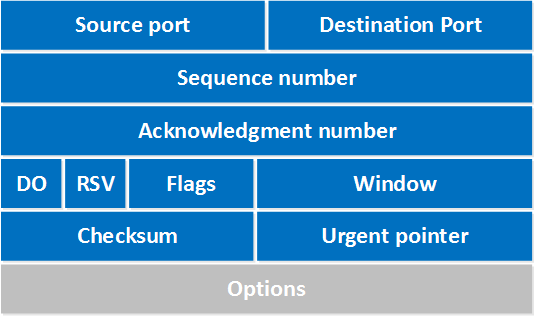
* **Version** – prvo polje govori o tome koja verzija IP se koristi. S obzirom na to da samo Ipv4 koristi ovo zaglavlje, ovde će uvek stojati broj 4.
* **Header Length** – ovo četvorobitno polje govori o dužini IP zaglavlja u 32-bitnim inkrementima. Minimalna dužina je 20 bajtova, stoga će stojati broj 5 ovde. Najveći broj koji ovde može da stoji 15 pa će to biti dužina od 60 bajtova. Ovo polje je takođe poznato i kao IHL (Internet Header Length).
* **Type of Service** – koristi se za QoS (Quality of Service). Zauzima 8 bitova koji se koriste da se označi paket da bi se paket posebno označio.
* **Total Length** – ovo 16-bitno polje označava veličinu celog IP paketa (zaglavlje plus podaci) u bajtovima. Tako da je najmanja dužina 20 bajova (ukoliko nema podataka), a najveća je 65535 bajtova, što je ujedno i najveći broj označen sa 16 bitova.
* **Identification** – Koristi se ako se radi fragmentacija. Tada će svi paketi koji se fragmenti ovde imati istu vrednost i na taj način će se reasemblirati na prijemu.
* **IP Flags** – takođe se koriste za fragmentaciju, ima ih 3:
  + Prvi bit je uvek postavljen na 0.
  + Drugi bit se zove DF (Don’t fragment) i označava da ovaj paket ne treba fragmentirati.
  + Treći bit se zove MF (More fragments) i setovan je na svim fragmentiranim paketima osim na poslednjem.
* **Fragment Offset** – ovo 13-bitno polje specificira poziciju fragmeta u originalnom fragmentiranom IP paketu.
* **Time to Live** – Svaki put kada IP paket prođe kroz ruter, ovo polje se dekrementira za 1. Kada dostigne 0, ruter će odbaciti paket i poslati ICMP poruku da je vreme isteklo pošiljaocu. Ovo polje ima 8 bitova i služi da onemogući da paketi zauvek kruže po mreži.
* **Protocol** – 8-bitno polje govori o tome koji protokol je enkapsuliran u sam IP paket. Tako na primer TCP ima vrednost 6, a UDP ima vrednost 17.
* **Header Checksum**: ovo 16-bitno polje služi da pamti checksum zaglavlja. Na prijemnoj strani se ovo polje može iskoristiti da bi se utvrdile eventualne greške unutar zaglavlja. Tipično greška može nastupiti pri prolasku kroz mrežu.
* **Source Address** – 32-bitna IP adresa izvora.
* **Destination Address** – 32-bitna adresa odredišta.
* **IP Option** – Ovo polje se retko koristi. Ima promenljivu dužinu zavisno od opcija koje se koriste. Kada se ovo polje koristi, vrednost u polju Header Length će se povećati. Na primer, moguće je koristiti opciju source route kada pošiljalac zahteva određenu putanju rutiranja.

Na sledećoj slici je prikazano realno IP zaglavlje korišćenjem alata Wireshark gde se vide gore opisana polja:



### TCP protokol

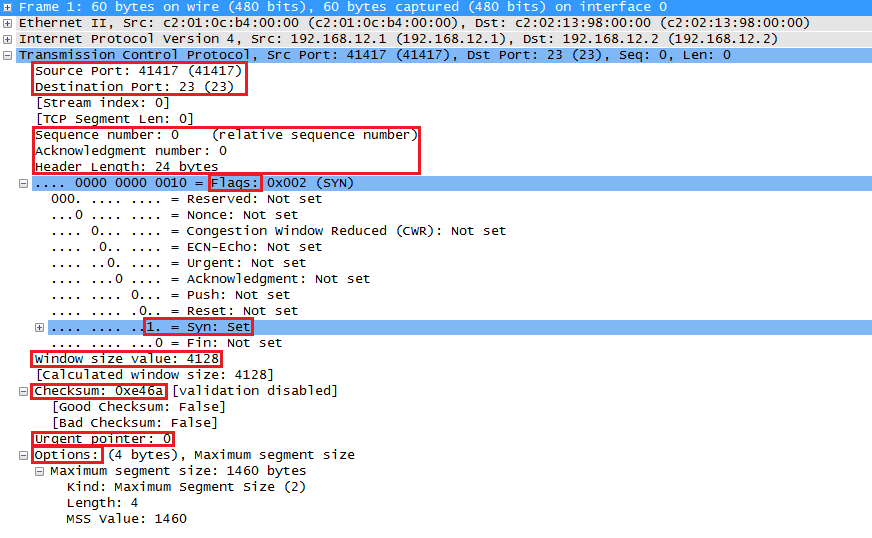
TCP (Transmission Control Protocol) je pouzdani transportni protokol pošto uspostavlja konekciju pre nego što pošalje bilo kakve podatke i sve što pošalje je potrebno da se potvrdi sa strane prijemnika. Na sledećoj slici je prikazano zaglavlje TCP paketa:



TCP zaglavlje se sastoji od sledećih polja:

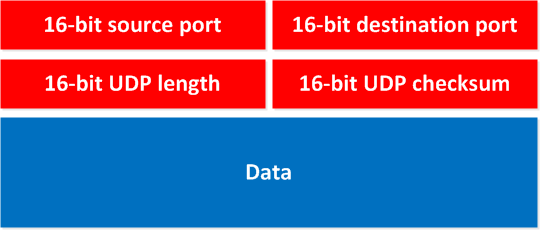
* **Source Port** – 16-bitno polje koje specificira broj porta pošiljaoca.
* **Destination Port** – 16-bitno polje koje specificira broj porta prijemnika.
* **Sequence Number** – 32-bitni broj koji označava koliko podataka je poslato za vreme TCP sesije. Kada se uspostavi nova TCP konekcija (tzv. 3 way handshake) tada je inicijalni sequence number postavljen kao nasumični 32-bitni broj. Prijemnik će koristiti ovaj sequence number i poslati nazad poruku prihvatanja (eng. acknowledgment).
* **Acknowledgment Number** – ova 32-bitna vrednost se koristi od strane prijemnika da zatraži sledeći TCP segment. Ova vrednost će biti sequence number inkrementirana za 1.
* **DO** – ovo je 4-bitno polje koje označava pomeraj odakle počinju podaci. Drugim rečima, govori o tome koliko je veliko zaglavlje (Data Offset).
* **RSV** – 3-bitno rezervisano polje. Za sada nije setovano, tj. postavljene su 3 nule.
* **Flags** – 9-bitno polje flegova, takođe se zovu upravljaći bitovi. Koriste se za uspostavljanje konekcije, slanje podataka i terminiranje konekcije:
  + **URG** – Urgent Pointer. Kada je ovaj bit postavljen, podaci u ovom paketu se tretiraju kao prioritetni u odnosu na podatke iz drugih paketa.
  + **ACK** – koristi se za odobravanje paketa.
  + **PSH** – predstavlja push funkciju. Govori aplikaciji da podaci trebaju biti poslati odmah i da nije potrebno čekati na popunjavanje celog TCP segmenta.
  + **RST** – služi za resetovanje konekcije. Kada pristigne kod odredišta, potrebno je terminirati konekciju. Koristi se samo kada se jave greške koje nije moguće prevazići i kada nije moguće normalno terminirati konekciju.
  + **SYN** – koristi se za inicijalni 3 way handshake i služi za inicijalno setovanje squence number polja.
  + **FIN** – koristi se za terminiranje TCP konekcije. TCP je full duplex tako da je potrebno da obe strane setuju ovaj fleg. Ovo je normalan način za terminiranje konekcije.
* **Window** – 16-bitno polje koje specificira koliko bajtova će ordredište primiti.
* **Checksum** – 16-bitno polje koje se koristi za proveru grešaka na prijemniku.
* **Urgent Pointer** – 16-bitno polje koje se koriti kada je setovan fleg URG. Označava gde se završavaju urgentni podaci.
* **Options** – Ovo polje je opciono i može biti između 0 i 320 bitova.

Na sledećoj slici je prikazano realno TCP zaglavlje korišćenjem alata Wireshark gde se vide gore opisana polja:



### UDP protokol

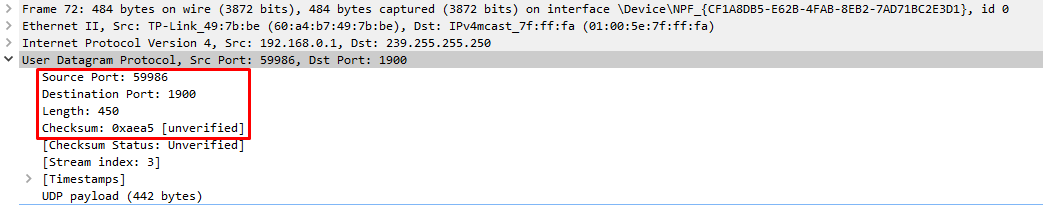
Zaglavlje UDP paketa je prikazano na sledećoj slici:



Kao što s vidi sa slike, zaglavlje je prilično jednostavno, a polja koja se tu nalaze označavaju sledeće:

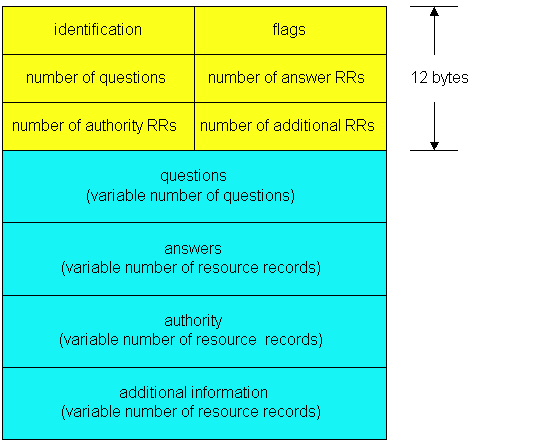
* **16-bit source port** – označava port izvora.
* **16-bit destination port** – označava port odredišta.
* **16-bit UDP length** – označava veličinu UDP paketa.
* **16-bit UDP checksum** – checksum koji se koristi na prijemnoj strani radi validiranja zaglavlja.

Na sledećoj slici je prikazano realno TCP zaglavlje korišćenjem alata Wireshark gde se vide gore opisana polja:



### DNS protokol

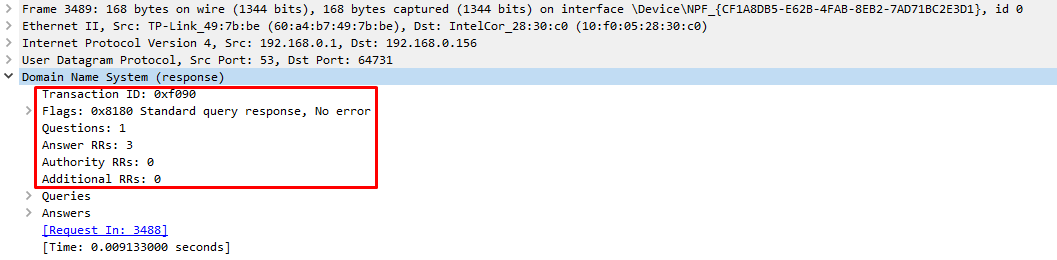
Zaglavlje UDP paketa je prikazano na sledećoj slici:



Prvih 12 bajtova predstavljaju zaglavlje DNS poruke, a ta polja označavaju sledeće:

* **Identification** – 16-bit broj za identifikaciju upita, odgovor sadrži isti broj.
* **Flags**:
  + 1bit: uipt (0) ili odgovor (1)
  + rekurzivni upit se traži
  + rekurzija podržana
  + 1 bit: odgovor je od autorizovanog name servera
* **Number of** – broj pojavljivanja 4 tipa podataka koji slede iza zaglavlja.

Na sledećoj slici je prikazano realno TCP zaglavlje korišćenjem alata Wireshark gde se vide gore opisana polja:

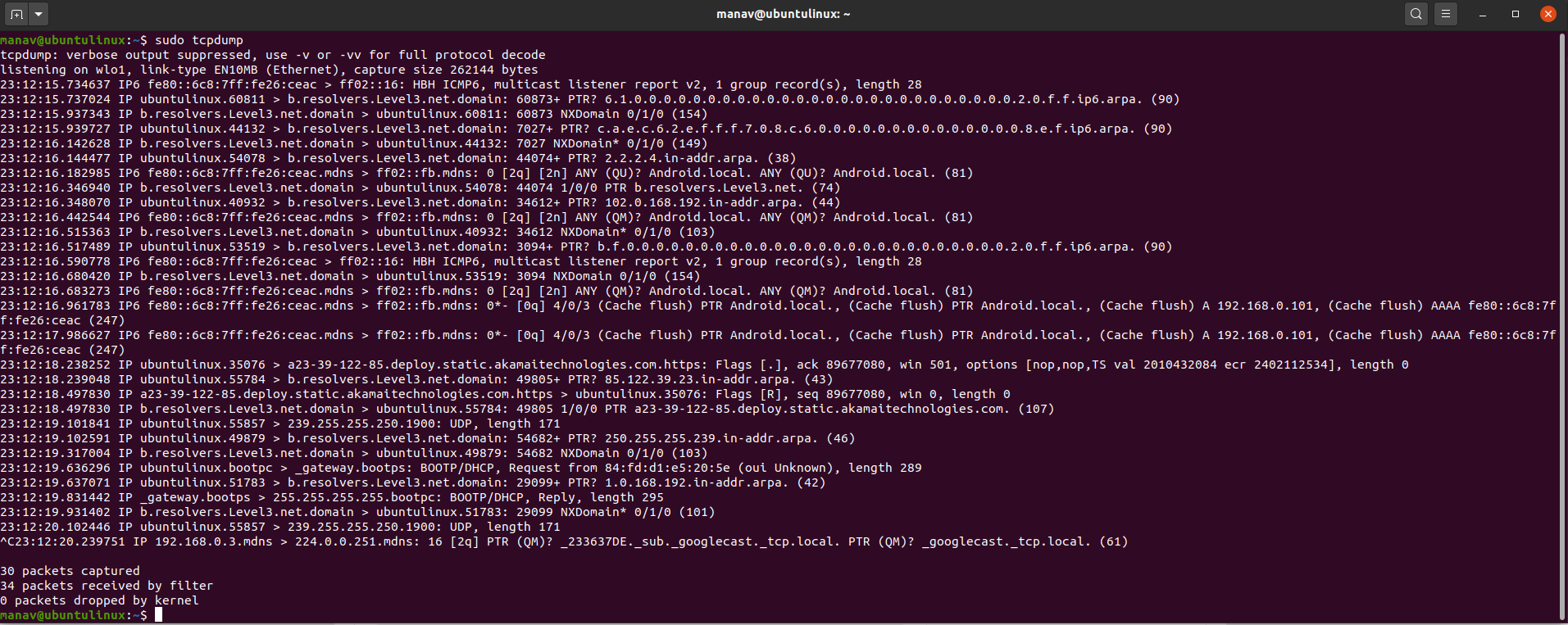


# Postojeća softverska rešenja

Do sada je razvijen veliki broj alata za mrežnu forenziku, a neki od značajnijih i češće korišćenijih će biti pomenuti u nastavku ovog rada.

## TCPDUMP

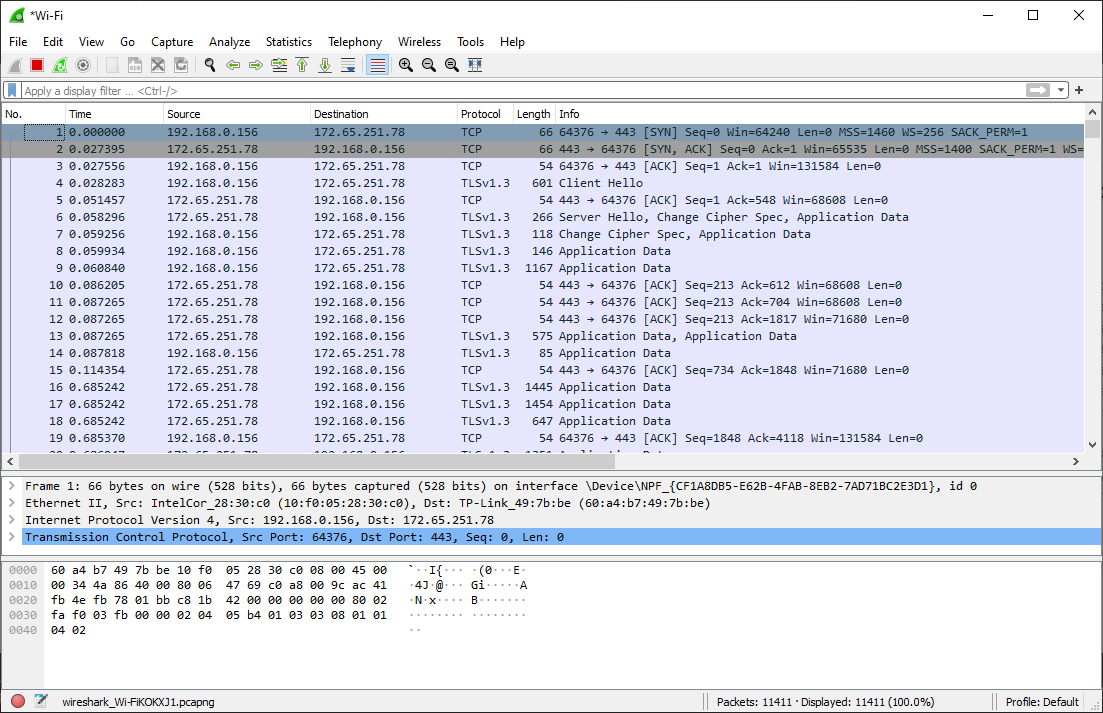
Ovaj alat predstavlja jedan od prvih razvijenih alata za potrebe mrežne forenzike. Ima mogućnost da sortira objekte po mrežnom saobraćaju. Takođe, ima mogućnost da identifikuje patterne. Radi preko command line interfejsa. Radi na Unix-like mašinama (Unix, Solaris, macOS), a postoji i port verzija za Windows mašine koja se zove WinDump. Na sledećoj slici je prikazan izgled izlaza nakon pokretanja ove aplikacije:



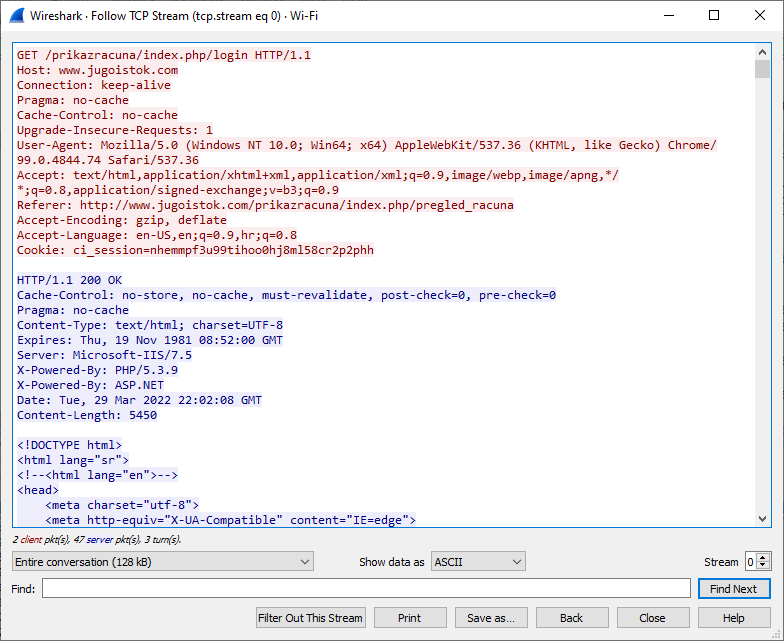
## Wireshark

Predstavnik je grupe tzv. sniffera. To su alati koji sa mrežnog adaptera prikupljaju sadržaj svih paketa na mreži (bit-by-bit). Naslednici su tcpdump projekta uglavnom. Najčešće koriste standardizovani format u koji snimaju prikupljeni sadržaj paketa (za eventualnu dalju analizu snimljenog saobraćaja). Tipičan primer formata je PCAP format. Podržava veliki borj opcija za filtriranje prilikom prikupljanja ili prilikom prikaza paketa. Ovaj alat poznaje strukturu PDU na svim nivoima TCP modela. Open source je projekat Wireshark fondacije.

Na sledećoj slici je prikazan prozor sa snimljenim mrežnim saobraćajem:



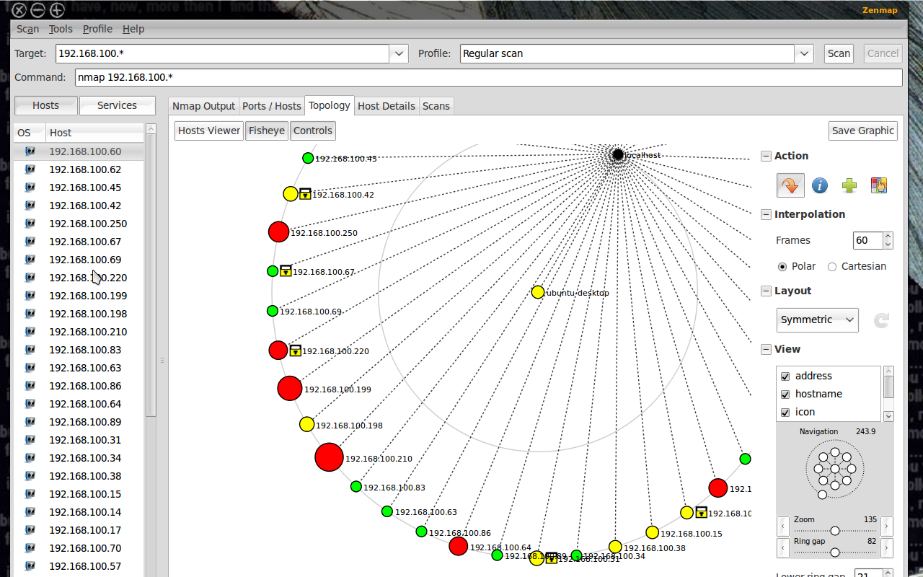
Na slici ispod je prikazan jedan TCP stream, gde je crvenom bojom prikazan sadržaj odlazećeg saobraćaja a plavom je prikazan sadržaj dolazećeg saobraćaja.



## NMAP (Zenmap)

Zenmap alat je GUI apliakcije NMAP. Ovi alati služe za mapiranje topologije računarskih mreža tako što skenira zadati opseg IP adresa šaljući pakete na neke od unapred poznatih portova. Zatim detektuje tip mrežnog uređaja na svakoj od IP adresa koje skenira. Ima mogućnost grafičkog prikaza otkrivene topologije mreže.

Na sledećoj slici je prikazan izgled jedne otkrivene mrežne toplogije pomoću ovog alata:



## NetWitness

Za razliku od Wireshar alata koji predstavlja sniffer alat, NetWitness pripada grupi alata koji se zovu analizatori saobraćaja. Fokusiraju se na sadržaj mrežnog saobraćaja i rade na nivou sesija, a ne na nivou paketa.

Moguće je importovati fajl sačuvan u Wireshark alatu i analizirati mrežni saobraćaj koji je već snimljen i pripremljen za analizu. Nakon toga je moguće prikazati analizu snimljenih podataka, kao i izveštaj koji sadrži broj sesija za svaki tip konekcije. Takođe je moguće rekonstruisati web stranice iz snimljenog saobraćaja.

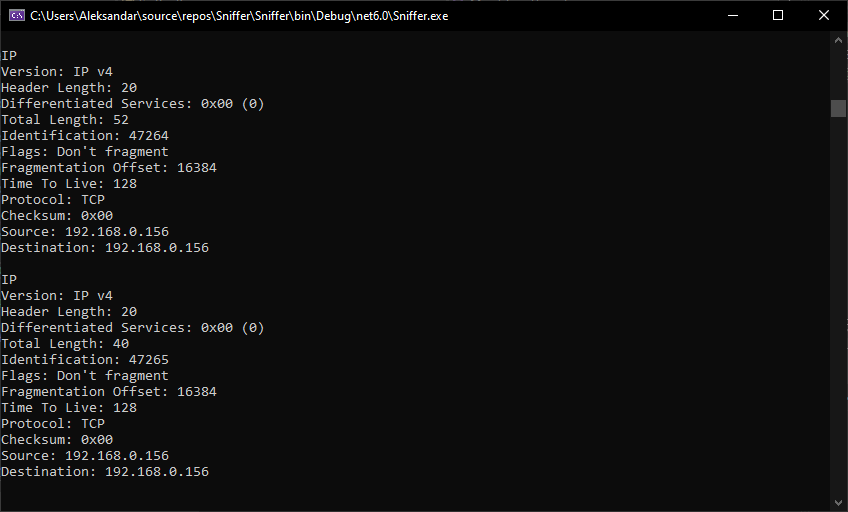
# Implementacija alata

Kao što je već i rečeno u uvodu ovog rada, implementacija alata je rađena u Microsoft .Net tehnologiji verzije 6.0. Reč je o konzolnoj aplikaciji koja ima mogućnost odabira interfejsa na kom će se snimati mrežni saobraćaj, zatim odabira filtera tj. mrežnih protokola koji se žele pratiti. Nakon odabira ovih parametara započinje snimanje mrežnog saobraćaja.

Na sledećoj slici je prikazana aplikacija u trenutku odabira mrežnog interfejsa i željenih protokola:

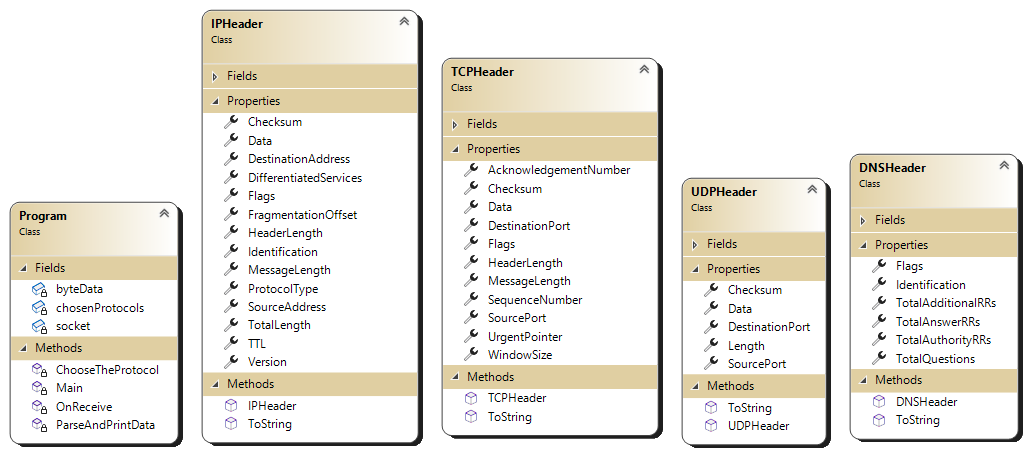


Nakon pritiska tastera Enter pokreće se snimanje mrežnog saobraćaja. Izlaz izgleda kao na slici ispod:



Kada se želi prestati sa snimanjem saobraćaja, neophodno je pritisnuti Control C i onda je moguće izlistati sve pakete kojin su snimljeni sa odgovarajućim zaglavljiva odabranih mrežnih protokola.

Na sledećoj slici su prikazane klase koje su implementirane:



* Klasa **Program** – sadrži Main metodu, i sldeće pomoćne metode:
  + **ChooseTheProtocol** – pomoćna metoda koja implementira odabir žejenih mrežnih protokola.
  + **OnReceive** – metoda koja se izvršava kada stigne paket koji je selektovan filterima.
  + **ParseAndPrintData** – pomoćna metoda koja je odgovorna za parsovanje sadržaja zaglavlja mrežnog protokola i njeno štampanje na standardni izlaz.
* Ostale klase koja implementiraju zaglavlja odgovarajućih mrežnog protokola. U svrhu čitanja podataka iz Socket-a koji se prima, koriste se klase MemoryStream i BinaryReader. To se obavlja u konstruktoru klasa. Propertiji svake klase predstavljaju podatake iz zaglavlja odgovarajućih mrežnih protokola. Metode koje implementiraju su:
  + **<*protocol*>Header** (konstruktor) – obavlja čitanje iz binarnog memory streama i setuje odgovarajuće propertije.
  + **ToString** – kreira odgovarajući string koji predstavlja reprezentaciju koja će se poslati na standardni izlaz.

# Literatura

[1] Hadoop predavanje, Kurs Računarstvo u oblaku, Elektronski fakultet u Nišu, Univerzitet u Nišu <https://cs.elfak.ni.ac.rs/nastava/pluginfile.php/7322/mod_folder/content/0/Hadoop-2019%2001.pdf?forcedownload=1>

[2] Euclidean vs Manhattan vs Chebyshev Distance <https://iq.opengenus.org/euclidean-vs-manhattan-vs-chebyshev-distance/>

[3] What is Hadoop? <https://databricks.com/glossary/hadoop>

[4] K-Means, The basic idea <https://stanford.edu/~cpiech/cs221/handouts/kmeans.html>

[5] K-Means Clustering Algorithm <https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning>