Sustavi za analizu mrežnog prometa

Dujak, Ena

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ena Dujak

SUSTAVI ZA ANALIZU MREŽNOG PROMETA

DIPLOMSKI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Ena Dujak

JMBAG: 0016129773

Studij: Baze podataka i baze znanja

SUSTAVI ZA ANALIZU MREŽNOG PROMETA

DIPLOMSKI RAD

Mentor:

Izv. prof. dr. sc. Ivan Magdalenić

Varaždin, svibanj 2022.

Ena Dujak

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Veća povezanost svijeta internetom sa sobom donosi i izazove kao što je konzistentna mrežna i informacijska sigurnost te je tema ovoga rada upravo analiza i korištenje sustava za analizu mrežnog prometa u svrhu pronalaska i iskorištavanja nesigurnosti u mreži. Kroz tijek rada tako će biti prikazana teorijska osnova iza njih te njihova upotreba na praktičnim primjerima koristeći virtualna računala povezana u internu mrežu jednostavne topologije. U praktičnome dijelu rada bit će kreirano nekoliko mogućih scenarija gdje će korištenjem sustava za analizu mrežnog prometa kao što je *Ettercap* biti izvršen napad na lokalnu mrežu. Osim napada na mrežu bit će prikazane i neke druge korisne radnje u kojima ovakvi sustavi mogu poslužiti. Jedan takav primjer bit će dan kroz Python program koji reaktivno prati na računalo pristigle pakete koristeći *tshark*. Prikazana je i funkcionalnost nekoliko alternativnih sustava za analizu mrežnog prometa kao što je *Capsa Portable Network Analyzer* te njihove prednosti i mane u usporedbi s iznimno popularnim *Wiresharkom*.

Ključne riječi: Mrežna sigurnost; Mrežni promet; Wireshark; Virtualizacija; Internet; Lokalna mreža; Ettercap; Man in the middle; Mrežni protokoli;

Sadržaj

1. Uvod

Mrežna sigurnost pojam je koji godinu za godinom postaje sve relevantniji u svijetu te se gotovo svaki dan na vijestima može čuti kako se ponovno dogodio hakerski napad. Činjenica je da veliki broj tvrtki ne ulaže dovoljno u mrežnu sigurnost dok ne bude prekasno. U mrežnu sigurnost potrebno je ulagati konstantno te raditi na tome da su zaposleni stručnjaci koji rade na svojim znanjima i vještinama te su proaktivni u korištenju novih tehnologija. Smatram da je mrežna i informacijska sigurnost vrlo važna pa ću u ovom diplomskog radu zbog toga govoriti upravo o tome; najboljim praksama u mrežnoj sigurnosti te o sustavima koji se koriste kako bi uzrokovali, ali i izbjegli potencijalne napade na mrežu.

Prethodno spomenuti sustavi upravo su fokus ovog rada te će nakon teorijske osnove na temelju koje funkcioniraju biti prikazano kako rade. Sustav za analizu mrežnog prometa na kojemu ću temeljiti ovaj rad te sve usporedbe bit će *Wireshark* koji je jedan od najpopularnijih takvih sustava te je ujedno i otvorenoga koda čime se još više ističe od drugih njemu sličnih sustava te je zbog toga meni posebno privlačan. U praktičnome dijelu tako ću, u izoliranom sustavu virtualnih računala, koristeći upravo *Wireshark* i druge njemu slične sustave izvesti nekoliko akcija kao što su *Man in the middle* napad te *sniffing* paketa ne bi li se došlo do privatnih podataka korisnika. Osim toga, smatram da je važno prikazati i kako ovakvi sustavi ne služe isključivo svrsi mrežne sigurnosti pa ću napisati i prikazati program u Pythonu čija je zadaća reaktivno praćenje mrežnih paketa na korisnikovom računalu. Program korisnik pokreće kada na tom računalu želi ograničiti pristup neželjenim internetskim stranicama te se koristeći *tshark* (*Wireshark* u komandnoj linij) paketi pristupljenih stranica prihvaćaju i analiziraju te se korisnika obavještava o pristupu nepoželjnim stranicama. Sustavi za analizu mrežnog prometa naime mogu se koristiti u velikom broju scenarija te smatram kako je važno i korisno znati ih koristiti.

Za pisanje rada koristit ću najviše internetske izvore kao što su dokumentacijske stranice pojedinih sustava i alata, ali i stručnu literaturu koja se tiče mrežne i informacijske sigurnosti te načina na koji funkcioniraju pojedini sustavi za analizu mrežnog prometa. Na samome kraju dat ću svoj osvrt na obrađenu temu i zaključke koje sam donijela u procesu.

2. Opis korištenih tehnologija

Budući da se radi o diplomskome radu fokusiranom na praktičnome znanju, korištene su metode i tehnike koje takav rad i inače odlikuju. Drugim riječima, korištene su istraživačke tehnike koje uključuju isprobavanje i pokazivanje tehnologija na stvarnim primjerima. Tehnologija korištena u radu može se podijeliti na *hardver* i *softver*, a što se tiče hardvera postavke računala su sljedeće:

- 16 GB RAM
- 512 GB SSD ROM
- Intel i7-7700HQ CPU - 2.80GHz procesor
- Killer Wireless kartica za Wi-Fi

Računala na kojima će biti izvršeni neki od spomenutih scenarija bit će virtualna kako bi se na najjednostavniji način simulirali željeni uvjeti. Dakle, od softvera korištene su sljedeće tehnologije:

- Windows 10 verzija 21H2 na primarnom računalu
- Oracle VM VirtualBox verzija 6.1.34
- Wireshark verzija 3.6.5
- Tshark
- Ettercap
- Capsa Portable Network Analyzer
- Python

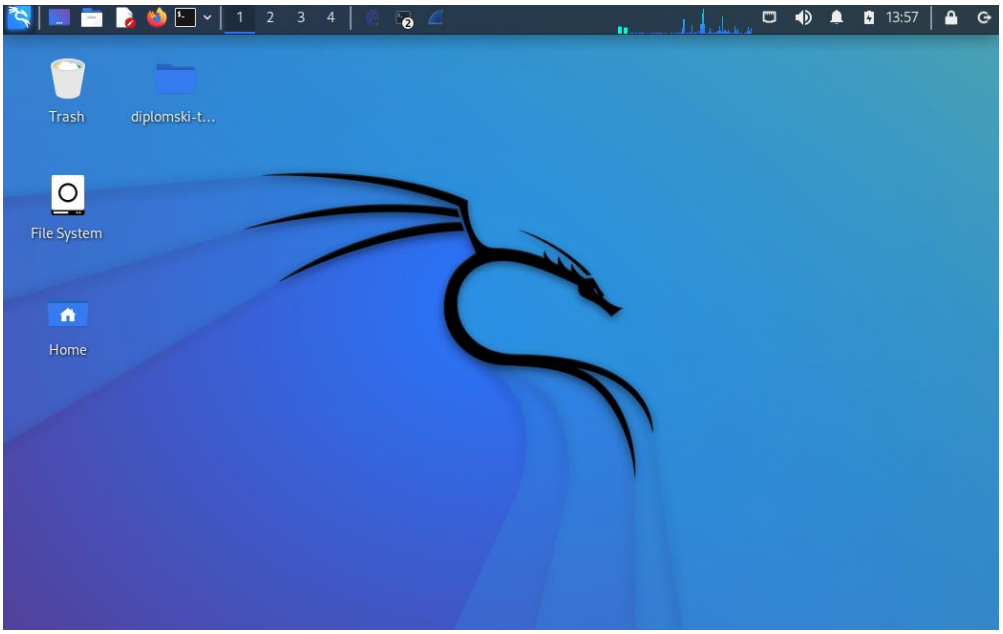
Oracle VirtualBox aplikacija je koja služi virtualizaciji te će u ovome kontekstu služiti kao polazišna točka za kreiranje scenarija te potrebnih postavki na računalima. Druge navedene aplikacije detaljnije će biti opisane u sljedećim poglavljima. Kreirana su tri virtualna računala na kojima su instalirani sljedeći operacijski sustavi:

- Ubuntu Server 22.04
- Ubuntu Desktop 22.04
- Kali Linux 2022.2

Kali Linux operacijski sustav poslužio je kao primarno virtualno računalo putem kojega je praćen i analiziran mrežni promet te su testirane sve korištene aplikacije. Ova su tri računala povezana NAT mrežnom tehnologijom što znači da imaju pristup na internet no na internetu nisu vidljiva kao samostalna računala već se njihova lokalna adresa prevodi u adresu *host* računala. Na takav se način postiže izoliranost pogodna testiranju, ali i normalan pristup internetu koji pogoduje preuzimanju softvera i drugim akcijama.

2.1. Kali Linux

Kali Linux *open source* je operacijski sustav baziran na Debianu te kao takav je potpuno besplatan ali i jednostavan za navigiranje budući da zbog svojih korijena dijeli mnoge sličnosti s najpopularnijom Linux distribucijom – Ubuntu operacijskim sustavom. Ipak, za razliku od Ubuntua Kali Linux specijalizirani je operacijski sustav čija je svrha korištenje za potrebe informacijske sigurnosti kao što su penetracijsko testiranje, računalna forenzika te razna istraživanja u računalnoj sigurnosti [19]. Ovaj operacijskih sustav zbog svoje namjene dolazi s već instaliranim mnogobrojnim aplikacijama od kojih su i Wireshark, Ettercap te drugi. S web stranice Kali Linuxa također je moguće preuzeti i gotovu sliku za pripremu virtualnog računala, čisti operacijski sustav ili neke druge varijante koje mogu pogodovati raznim korisnicima. U ovome radu bit će korišten Kali Linux postavljen na virtualnom računalu pa su zbog toga ove mogućnosti posebno korisne; na sljedećoj slici prikazana je radna površina:



Slika 1: Kali Linux (izvor: slika autorice)

Kroz pisanje ovoga rada svrha će mu biti izvršavanje određenih testiranja te izvođenje svih napada u praktičnome dijelu rada.

3. Osnove teorija računalnih mreža

Shvaćanje sustava za analizu mrežnog prometa, kao i sve drugo, počinje postavljanjem temelja. Temelji u slučaju sustava za analizu mrežnog prometa analogni su temeljima teorije mreža – logično je iz toga zaključiti dakle da je potrebno prvo pojasniti pojmove iz teorija mreža kako bi se mogli pojasniti i sustavi i alati koji se koriste za njihovu obradu u praktičnome smislu. U ovome će poglavlju tako biti opisani pojmovi kao što su mrežni paketi i mrežni protokoli.

3.1. Mrežna arhitektura

Mreža računala sastoji se od klijentskih i poslužiteljskih računala koja su međusobno povezana usmjernicima i prespojnicima. Poslužiteljska računala služe adresiranju računala koristeći DNS i DHCP odnosno *Domain Name System* te *Dynamic Host Configuration Protocol*, između ostalog. DNS jest sustav za imenovanje računala u mreži te rezoluciju dodijeljenih IP adresa u čemu sudjeluje upravo DHCP [1]. Kada poslužiteljska računala ostalim računalima u mreži dodijele adrese te imena pomoću kojih ih druga računala mogu adresirati tada računala mogu razmjenjivati informacije u svojoj lokalnoj mreži ili na internetu s udaljenim računalima, ukoliko su s internetom povezana usmjernikom.

Kako bi bila funkcionalna i drugima razumljiva računalna mreža mora imati svoju arhitekturu – mrežna arhitektura tako opisuje niz pravila, standarda i protokola koji se unutar neke mreže koriste [1]. Jedna takva arhitektura jest i općepoznata ISO/OSI arhitektura koja se sastoji od 7 slojeva [2]:

7. Aplikacijski
6. Prezentacijski
5. Sesijski
4. Transportni
3. Mrežni
2. Podatkovni
1. Fizički

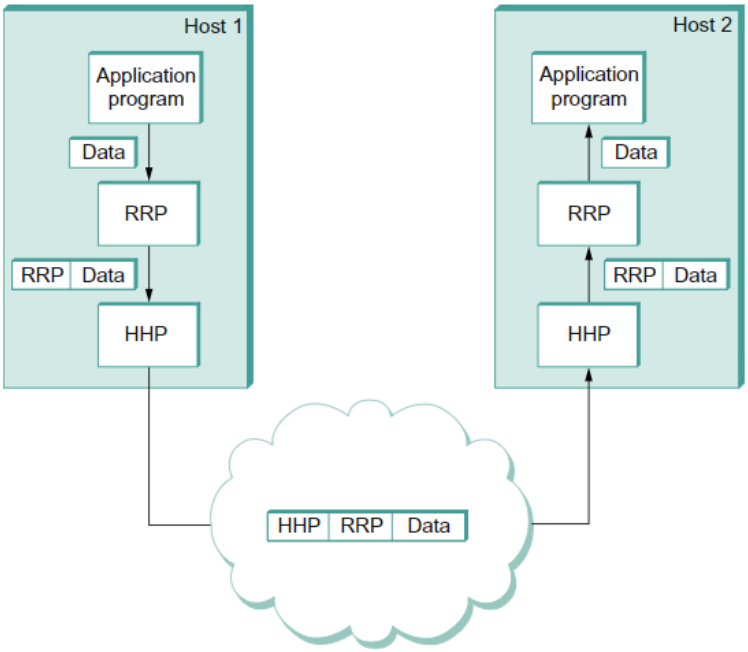
Svaki od ovih slojeva nosi se s različitim razinama apstrakcije. Prema [2] prvi se sloj nosi sa fizičkim potrebama mreže te mu nije bitno što preneseni podaci znače i što jesu jer se o tome brinu slojevi iznad toga. Podatkovni, transportni i mrežni brinu se o direktnom prijenosu podataka i načinu na koji će se podaci prenijeti. Najviša tri sloja apstrakcije služe pri kreiranju i upravljanju sesije, načinu na koji će se podaci na računalu „razumjeti“ te na kraju omogućavaju manipulaciju tim podacima kroz aplikacijski sloj.

3.1.1. Protokoli

Ovakvo raslojavanje zadaća u jednoj mreži omogućilo je nastanak nove ideje i danas potpuno neophodno korištenje protokola. Naime, ISO/OSI model nastao je prije vremena protokola i danas se kao takav ne koristi već služi samo za lakše razumijevanje mrežnih arhitektura te za razumijevanje čemu protokoli kao takvi uopće služe te zašto ih se danas toliko koristi. U ISO/OSI arhitekturi vidljivo je da postoje apstrahirani mrežni slojevi od kojih se svaki brine o jednom dijelu zadaće prilikom razmjene, pohrane i na kraju manipulacije podataka. Niži slojevi koji se brinu o samoj razmjeni podataka, kao što je već navedeno, ne moraju znati o kakvim se podacima radi i koja je njihova krajnja svrha. Isto tako, viši slojevi koji se brinu o prikazivanju podataka ne moraju znati kako su ti podaci došli do računala i na koji način. Ovaj princip primjenjuje se i na korištenje protokola čija je glavna zadaća sakrivanje svih zadaća osim onih kojima se određeni dio mrežne arhitekture bavi [1]. Korištenje protokola u nekoj mreži olakšava posao svima uključenima jer omogućava fokusiranje upravo na onaj dio mreže koji je korisniku potreban. Mrežnim inženjerima bit će važno znati na koji se način i kroz koje kanale podaci prenose od računala do računala dok s druge strane razvojnim inženjerima isto to neće biti važno, već trebaju samo znati mogućnosti prenošenja podataka kojima zatim može manipulirati kako bi razvio web aplikacije ili jednostavno slao e-mailove. Prema tome, može se reći kako protokoli omogućavaju i upravo čine slojeve mreže te omogućavaju uvid u i manipulaciju upravo onog dijela procesa prijenosa podataka koji je korisniku potreban [1].

Kako bi takva mreža uopće bila moguća potrebno je podatke pripremiti kako bi svaki protokol znao o kojoj se vrsti apstrakcije radi. Iz teorija mreža poznato je i kako se podaci ne prenose u svojoj cijelosti već u obliku velikog broja manjih paketa koji zatim moraju imati svoj *header* ili *trailer* čija je dužnost identificirati i proslijediti *payload* odnosno paket koji prenose [1]. Ovdje se pojavljuje novi pojam – enkapsulacija. Enkapsulacija ponovno nastaje iz principa raslojavanja mrežne arhitekture te omogućava protokolima da prilikom pripreme paketa za slanje na isti postave svoj *header* koji ujedno služi i kao identifikator i tako slijedom od najviše do najniže razine [1]. Nakon što paket dođe do odredišnog računala identificira se prvi *header* koji se ujedno odnosi na najnižu razinu, taj se *header* miče te se paket

prosljeđuje do iduće razine gdje se ponovno odvija isti proces sve dok podatak ne dođe do programa kojemu je namijenjen [1]. Na sljedećoj slici preuzetoj iz [1] prikazana je enkapsulacija jednog paketa na primjeru:



Slika 2: Enkapsulacija (Izvor: [1])

Dakle, u gornjem lijevom kutu gdje je prikazano prvo računalo vidljivo je kako aplikacija želi poslati paket drugom računalu. Paket će prvo enkapsulirati koristeći protokol najviše razine koji u ovome slučaju identificira aplikaciju kojoj se šalje te istoj prosljeđuje podatak. Vidljivo je također i kako sada paket sadržava *header* tog istog protokola te paket nastavlja do sljedećeg protokola koji je potreban kako bi se paket ispravno poslao do drugog računala. Nakon što se paket ponovno enkapsulira i *headerom* ovog protokola takav se šalje drugom računalu gdje se proces odvija u obrnutom redoslijedu. Iako se u ovome slučaju radi o protokolima koji su zadani samo kao primjer, neki od protokola koji su u širokoj upotrebi jesu HTTP/S, FTP, SMTP, TCP, UDP i drugi. TCP i UDP posebno su važni protokoli budući da se njima koristi veliki dio interneta kakvog koristimo danas. TCP je kratica koja znači *Transmission Control Protocol*, dok UDP znači *User Datagram Protocol* – TCP je nešto sporiji od UDP-a budući da se brine o tome da je svaki paket došao na odredište i da je datoteka prenesena sa svojeg izvora na odredište u cijelosti, dok se UDP o tome ne brine već mu je jedina zadaća da pošalje aktualni paket [1]. TCP se tako primjerice koristi u prijenosu datoteka, dok se UDP uglavnom koristi za strujanje audio ili video sadržaja jer je u tom slučaju bitnija brzina prenošenja, a ne toliko točnost.

Iako na početku prikazana ISO/OSI arhitektura ne poznaje protokole i prikazuje vrlo rudimentarnu verziju mrežne arhitekture, nešto novija TCP/IP arhitektura je ta koja se

zapravo koristi na internetu te je naziv dobila po dvama protokolima koji se na internetu učestalo koriste [1]. Ova je arhitektura nešto jednostavnija te opisuje 4 sloja koji se mogu objasniti kao „sendvič“ gdje pecivo predstavlja, u prenesenom smislu, aplikaciju i samo računalo između kojih se nalazi niz protokola koji omogućavaju uspješno razumijevanje pridošlih paketa. Sustavi za analizu mrežnog prometa djeluju upravo u slojevima između tzv. peciva na način da analiziraju novopridošle pakete te korisniku omogućavaju uvid u *header* ili *trailer* svakog paketa, protokole na koje se odnose te sam *payload* odnosno podatak koji se u paketu prenosi. Očigledno je da se iz toga može izvući puno zanimljivih, a u nekom slučaju i tajnih podataka (što će biti prikazano u praktičnome dijelu rada). Osim toga, u teorijskom dijelu koji slijedi bit će prikazane sve funkcionalnosti sustava za analizu mrežnog prometa u kontekstu teorije mreža.

4. Sustavi za analizu mrežnog prometa

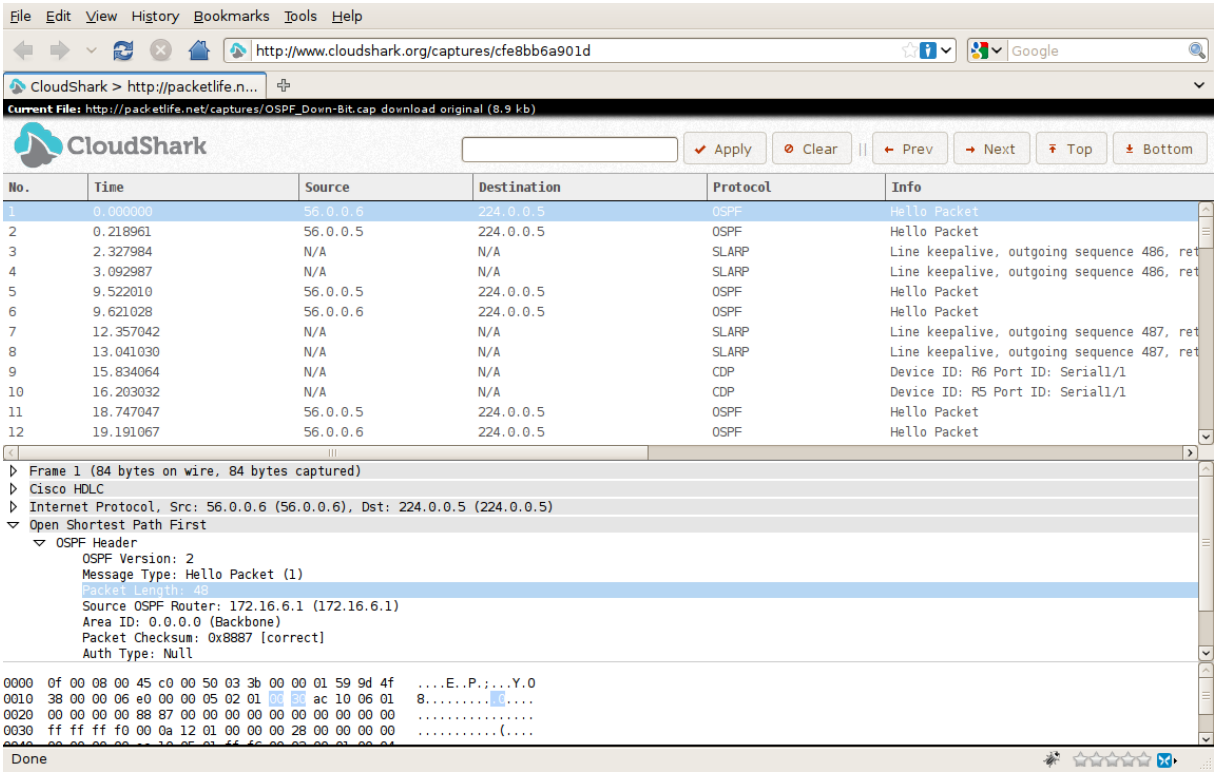
U prethodnome poglavlju opisana je teorija koju je potrebno razumjeti kako bi bilo jasno što je zapravo zadaća sustava za analizu mrežnog prometa. Prema [3] njihova je glavna zadaća analiza dolaznih i odlaznih paketa na računalo putem mreže te njihova disekcija po protokolima i zadaći. Važno je napomenuti da ovakvi programi ni na koji način ne utječu na nadolazeće ili odlazeće pakete već samo služe za njihovu analizu. No postoje i programi kao što je *Ettercap* koji osim osnovnih zadaća sustava za analizu mrežnog prometa ima i mogućnost izvođenja *Man in the middle* napada što će biti prikazano u zasebnome poglavlju.

Ipak, zasigurno najveći i najpopularniji odabir mrežnih inženjera upravo je *Wireshark* koji će biti opisan kao osnovni alat s kojim će biti uspoređeni drugi, njemu slični sustavi. Osim Ettercapa i Wiresharka još neki od popularnih odabira jesu [4]

- Capsa Portable Network Analyzer
- CloudShark
- Colasoft Capsa
- Sysdig
- Mojo Packets
- SolarWinds RMM

i drugi. Svi navedeni alati imaju grafičko sučelje te im je glavna zadaća upravno vizualizacija mrežnog prometa, a alati kao što je Mojo Packets imaju mogućnost generiranja grafova i razne druge mogućnosti vizualizacije mrežnog prometa – ili putem datoteke u koju su spremljeni prethodno „uhvaćeni“ podaci ili u stvarnom vremenu s tek dobivenim podacima. Dakle, u situaciji kada se radi o posebnom slučaju ili o slučaju koji se želi kasnije više analizirati, dohvaćeni podaci mogu se spremiti u datoteku s nastavkom .pcap ili novije .pcapng koje sustavi za analizu mrežnog prometa međusobno mogu otvarati i razumjeti. Tako je primjerice moguće dohvaćati mrežni promet koristeći CloudShark te iste te podatke poslije analizirati Wiresharkom. Osim ovih postoje još neki standardni formati kao što su: .wcap, .cap, .pkt i .libcap. Najviše se ipak koriste .pcap i .pcapng formati budući da se standardno koriste u Wiresharku.

Na sljedećoj slici prikazano je grafičko sučelje CloudShark sustava koja prikazuje samo neke od mogućnosti koje nudi:



Slika 3: CloudShark prikaz GUI-a (izvor: [5])

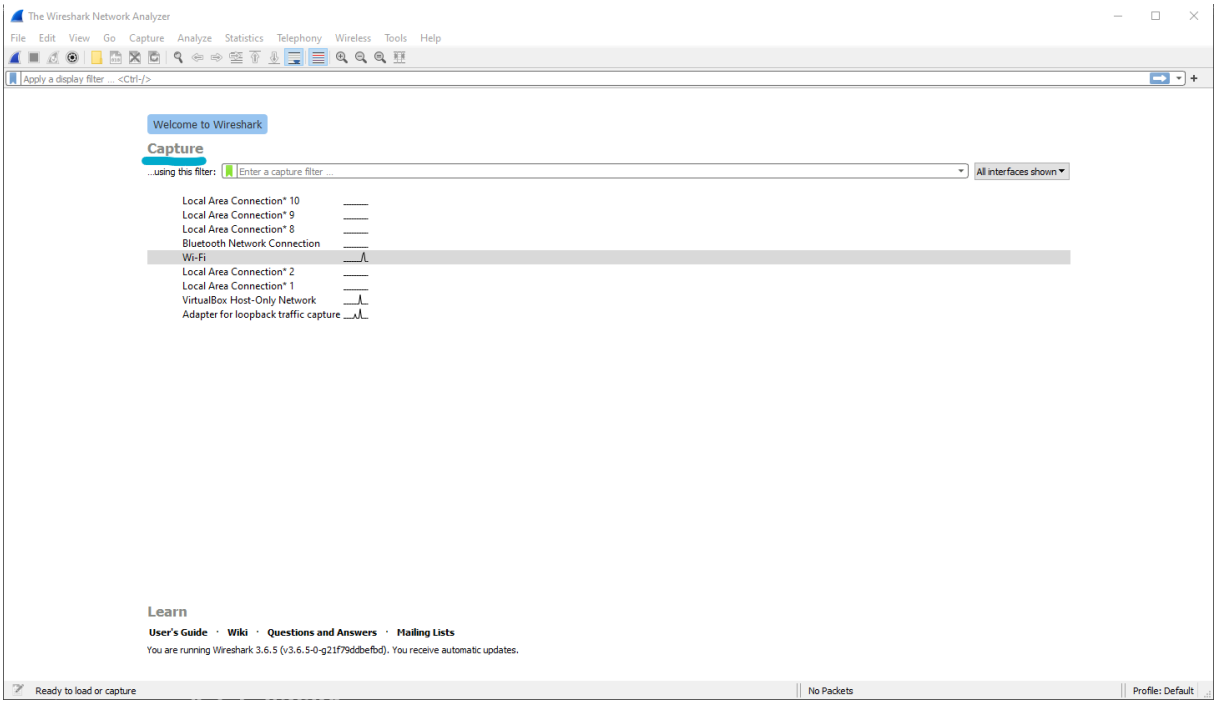
Osim osnovnih informacija kao što su broj paketa i vrijeme dolaska od početka analize, vidljivi su i podaci kao što su izvorišna i odredišna IP adresa te korišteni protokol i neke druge dodatne informacije. Ovakvo je grafičko sučelje klasično i odlika je većine sustava za analizu mrežnog prometa, pa će tako u idućem poglavlju kada će detaljnije biti obrađen Wireshark biti vidljivo koliko je njegovo grafičko sučelje zapravo slično ovome; što nije niti čudno budući da je Wireshark *de facto* industrijski standard. Već je iz ovih osnovnih podataka moguće otkriti informacije koje su korisniku inače tajne, kao što je izvorišna IP adresa i korišteni protokoli. U donjem je prozoru vidljivo više informacija o paketu no o tome će više riječi biti u sljedećem poglavlju.

Dobar dio sustava za analizu mrežnog prometa ima i svoja alternativna rješenja kroz iskazivanje podataka u komandnoj liniji. Za Wireshark to je tshark, a njemu slični tcpdump niti nema grafičko sučelje već je namijenjeno za korištenje samo u Linux terminalu. Ovakva se rješenja često ugrađuju u skripte na računalnim programima (ili u slučaju tsharka i u Python skriptama) gdje mogu imati razne zadaće. Jedan takav program bit će prikazan u praktičnome dijelu rada. Sada kada je postavljena teorijska osnova mrežne arhitekture te objašnjen način na koji sustavi za analizu mrežnog prometa rade, važno je osvrnuti se na najpopularniji takav sustav – Wireshark – te pomoću njega i u praktičnome dijelu prikazati čemu ovakvi sustavi mogu poslužiti.

4.1. Wireshark

Kao član obitelji otvorenoga koda Wireshark uživa popularnost ne samo mrežnih inženjera već i studenata, učenika i svih onih koji žele znati i naučiti nešto o području računalnih mreža. Unatoč tome što se radi o besplatnom alatu, Wireshark pruža veliku lepezu mogućnosti kao što su: dohvaćanje podatkovnih paketa putem raznih medija (WiFi, Ethernet, Bluetooth) i to na svim mrežnim sučeljima računala, otvaranje postojeće datoteke kreirane u drugim sustavima za analizu mrežnog prometa [3]. Wireshark također pohranjuje nove sesije i otvara postojeće, omogućuje razne statistike i uvide u mrežni promet, ali i detaljiziranje svakog zasebnog paketa po protokolima [3]. Na takav način i kroz svoje mogućnosti Wireshark omogućava uvid u sve što se događa na računalu prilikom, primjerice, strujanja YouTube videa ili slanja e-mailova. Raščlanjuje sve poslane i zaprimljene pakete na način da su razumljivi korisniku pa je na ovaj način moguće pratiti promet kako bi se uočile potencijalne opasnosti ili drugi sumnjivi promet. U praktičnome dijelu rada bit će prikazan rad starijih protokola (koji su i danas na dobrom dijelu servisa) te će se putem Wiresharka pokušati otkriti lozinke i drugi tajni podaci.

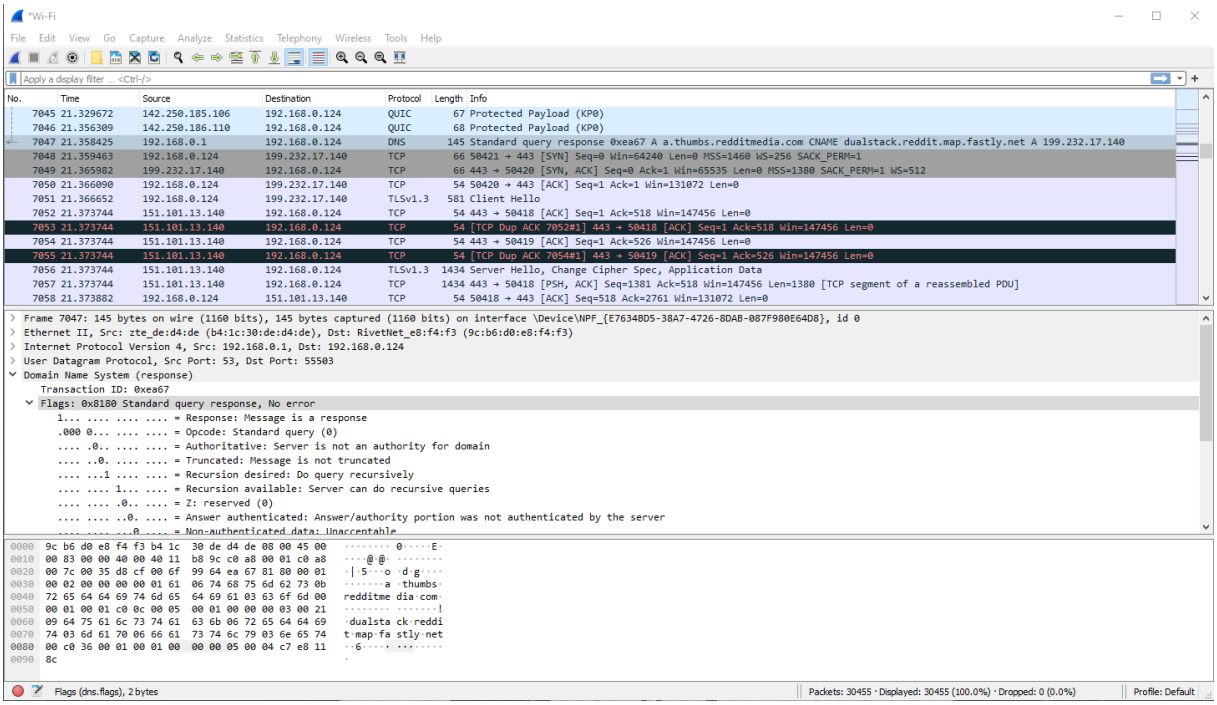
Početno sučelje Wiresharka vrlo je jednostavno te prikazuje dostupna sučelja nad kojima može dohvaćati pakete, a omogućuje i upis filtera ukoliko korisnik želi analizirati samo određene pakete odnosno protokole:



Slika 4: Wireshark - početni zaslon (izvor: slika autorice)

Koristeći animaciju otkucaja srca, Wireshark prikazuje trenutačno aktivna sučelja kako bi korisnik lakše mogao odrediti nad kojim želi vršiti analizu prometa.

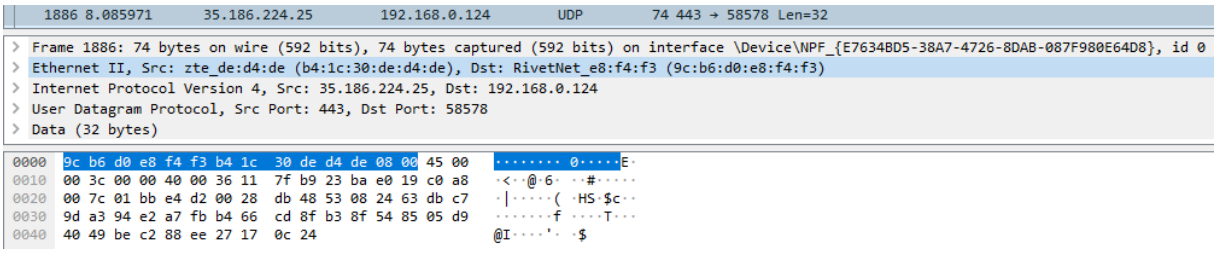
U ovome je slučaju odabrano Wi-Fi mrežno sučelje kako bi se moglo prikazati protokole i pakete koji trenutačno struje na *host* računalo. Na sljedećoj slici vidljivo je grafičko sučelje za dohvaćanje paketa te je očito kako je slično onome prikazanom na slici 3 i da se sastoji od vrlo sličnih elemenata:



Slika 5: Wireshark GUI (izvor: slika autorice)

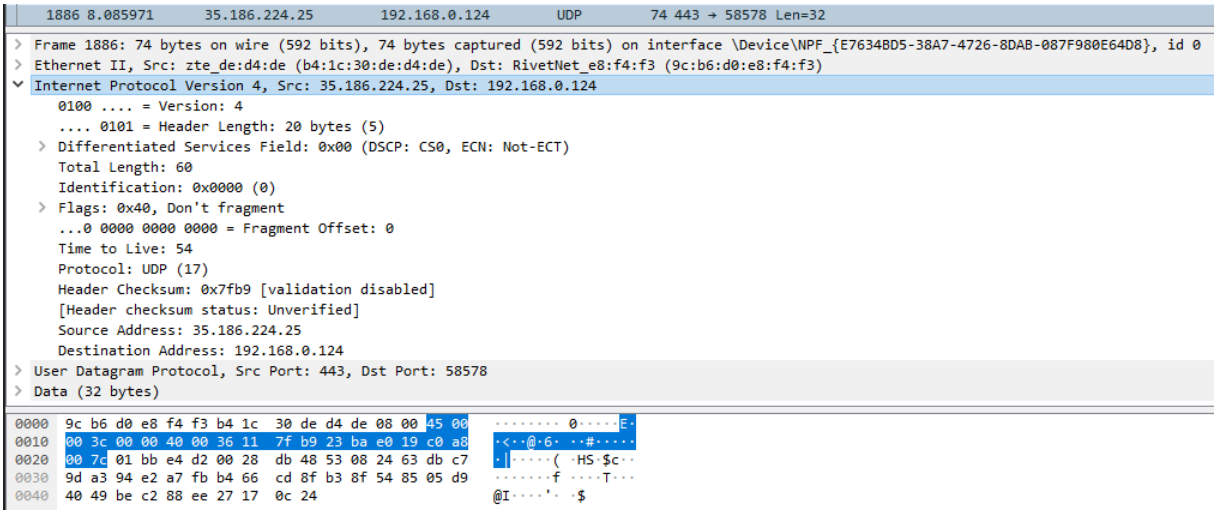
U stupcima su redom vidljivi: redni broj paketa, vrijeme dohvaćanja, polazišne i odredišne IP adrese, korišteni protokol, veličina paketa te dodatne informacije. Na ovoj je slici vidljivo nekoliko različitih protokola od kojih su najzanimljiviji Googleov novi protokol QUIC te DNS. Budući da je u vrijeme analize na računalu bio pokrenut YouTube video Wireshark je dohvatio QUIC pakete koje YouTube odnedavno koristi, a radi se o protokolu koji utjelovljuje (u teoriji) najbolje od UDP-a i TCP-a [6]. Naime, TCP nizom „rukovanja“ osigurava da će svi paketi od kojih se neka datoteka sastoji biti dostavljeni na odredište, dok je UDP-u glavna zadaća da pakete dostavi bez obzira na to je li neki paket u međuvremenu izgubljen. Ideja QUIC protokola je da uz osiguranje koje nudi TCP brzinom UDP-a omogući brzo ali i kvalitetno strujanje sadržaja [6]. DNS protokol u ovome je slučaju dohvaćen zbog otvaranja stranice reddit.com gdje *host* računalo od svojeg DNS poslužitelja traži rezoluciju poslužitelja i IP adresu na kojoj ovu stranicu može pronaći, nakon čega dolazi do niza TCP enkapsuliranih paketa u tijeku učitavanja stranice. Također je među dohvaćenim paketima vidljiv i TLS protokol i to njegova treća verzija, što je odlika danas standardnog HTTPS protokola kojeg koristi većina stranica na internetu. HTTPS koristeći SSL/TLS protokole omogućava enkripciju podataka te poboljšava sigurnost web stranica te onemogućuju korisniku izravan pristup podacima kroz TCP ili UDP.

Na sljedećih nekoliko slika prikazana je analiza jednog paketa.



Slika 6: Analiza paketa - 1. header (izvor: slika autorice)

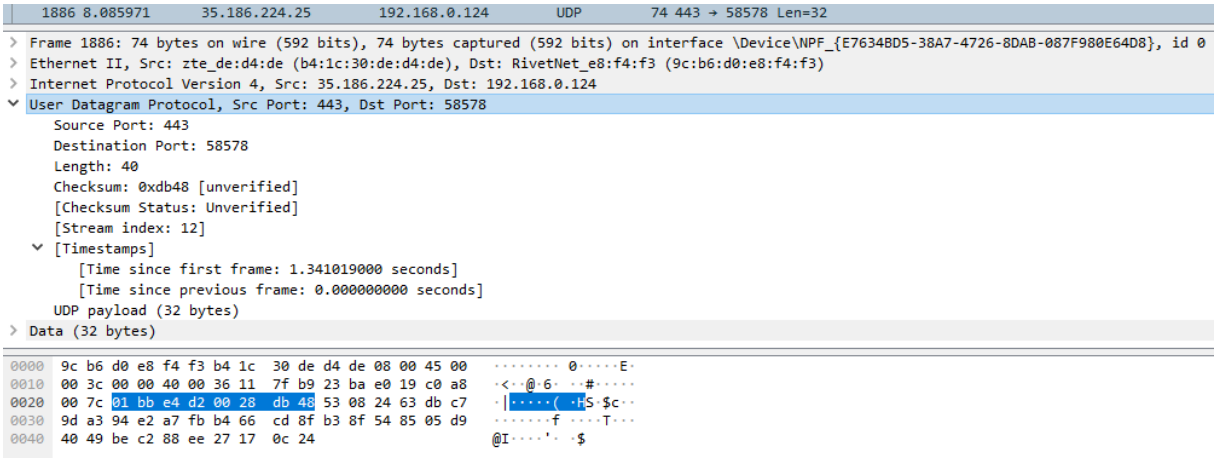
Na slici 6 prikazan je jedan paket čiji je primarni protokol UDP. Na *host* računalu se u vrijeme dohvaćanja paketa putem Spotify aplikacije struji glazba zbog čega je poželjno koristiti upravo UDP protokol nad TCP-om, zbog brzine prenošenja paketa. Ovaj paket, koji sadrži *payload* veličine 32 bajta kako je navedeno u posljednjem stupcu („Len = 32“), na sebi sadrži tri *headera* ili zaglavlja. Odmah ispod njega vidljiv je naslov „Frame“ koji označava cijeli paket i vidljivo je kako je ukupne veličine od 74 bajta. Ovaj paket dakle sadrži zaglavlja koja su ukupno veća od samog podatka koji prenose. Nadalje, drugi podnaslov i ujedno prvo zaglavlje odnosi se na fizički sloj mrežne arhitekture te se na slici 5 također može vidjeti što zaglavlje sadrži. Proširenjem zaglavlja može se vidjeti i informacija o sljedećem protokolu odnosno sloju mrežne arhitekture koji slijedi – u ovome slučaju to je IP protokol.



Slika 7: Analiza paketa - 2. header (izvor: slika autorice)

Na slici 7 vidljivo je drugo zaglavlje koje se odnosi na IP protokol čija je zadaća upravo adresiranje paketa te prenošenje informacije odakle je paket došao. U ovome se slučaju radi o još uvijek korištenom IP protokolu verzije 4, a njegova je veličina 20 bajtova te sadrži i TTL – *time to live* informaciju. Prema [1] TTL označava broj skokova koje paket može obaviti prije nego što se obriše. Ovo je korisno u slučaju kada primjerice izvorišno računalo na odredišno računalo pošalje paket no u tom trenutku odredišno računalo izgubi vezu s izvorišnim računalom. Kako izvorišno računalo ne bi beskonačno puta pokušavalo uspostaviti vezu s odredišnim koristi se TTL kao svojevrsni brojač koji određuje kada se paket prestaje pokušati slati. Nadalje, vidljiv je sljedeći protokol koji je u ovome slučaju UDP

zatim izvorišna i odredišna adresa računala. Još jedna zanimljiva informacija koja se može iščitati iz IP protokola jesu zastavice; u ovome slučaju postavljena je zastavica koja zabranjuje daljnju fragmentaciju paketa na disku.



Slika 8: Analiza paketa - 3. header (izvor: slika autorice)

Na slici 8 prikazano je i posljednje uključeno zaglavlje koje se odnosi na UDP protokol. Osim informacija o odredišnom i izvorišnom portu opisana je i veličina od 40 bajtova. Naime, UDP ima fiksnu veličinu od 8 bajtova [7], što zajedno s originalnih 32 bajta *payloada* čini ukupno 40 bajtova. Zajedno s 20 bajtova IP protokola preostaje da je na Ethernet zaglavlje potrošeno 14 bajtova. Nadalje, u *checksum* podnaslovu piše kako paket nije provjeren. *Checksum* je algoritam za provjeru ispravnosti paketa te u IP protokolu verzije 4 nije obavezan [1], a prema ovome nije niti korišten što nadalje odgovara opisanom djelovanju UDP protokola koji ne provjerava ispravnost poslanih paketa već mu je jedina zadaća njihovo slanje. Posljednji podnaslov opisuje vrijeme proteklo od slanja prvog paketa ukupne datoteke te vrijeme proteklo od slanja prethodnog paketa ukupne datoteke. Naslov *Data* veličine 32 bajta originalni je *payload* te ne sadrži nikakve dodatne informacije osim podatka koji je trebao biti poslan.

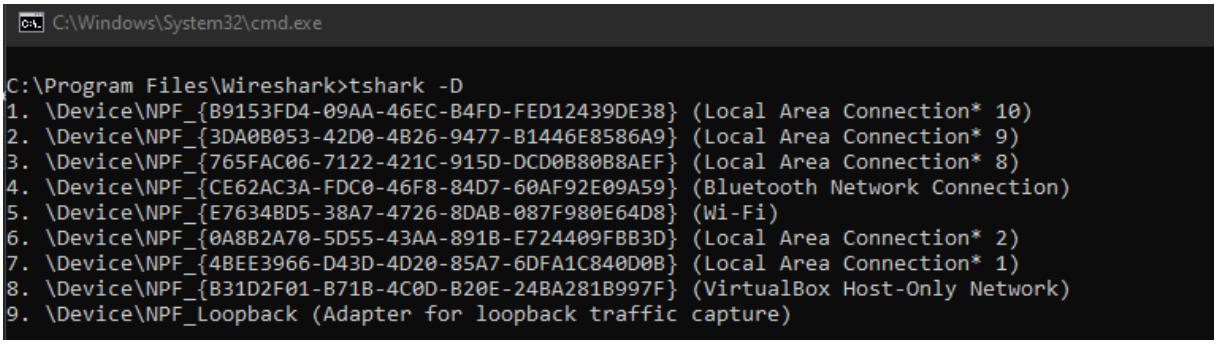
Na prethodno opisanom primjeru u praksi je prikazana izvedena enkapsulacija paketa koji od najnižih protokola kao što je Ethernet dolazi do IP protokola koji adresira paket te na kraju i do UDP-a koji odredišnom računalu daje tražene podatke. Ovakav je način prenošenja paketa reprezentativan za svakodnevno slanje i preuzimanje informacija na internetu.

4.1.1. tshark

Wireshark je definitivno vrlo korisni alat koji nudi pregršt informacija do u najmanje detalje. Ipak, u nekim situacijama sve te informacije prikazane u lijepom grafičkom sučelju nisu potrebne i ponekad je mrežnim inženjerima važno pratiti što se događa na mreži kroz komandnu liniju, u obliku servisa koji stalno radi u pozadini ili ugrađeno u kod aplikacije. Za

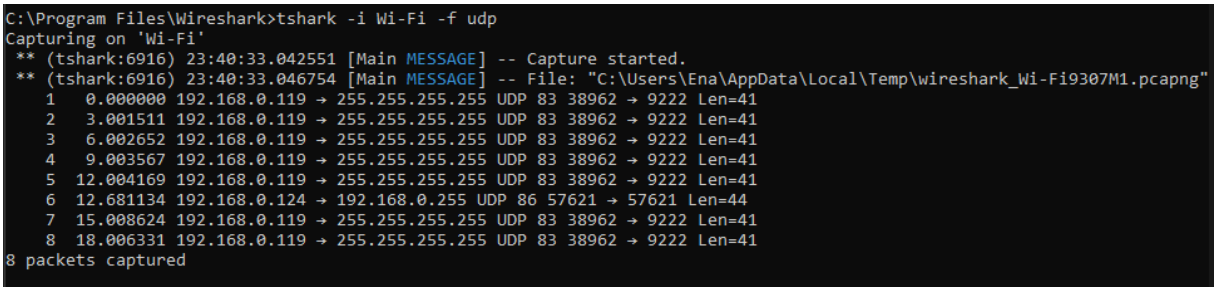
takve potrebe koristi se tshark, verzija Wiresharka u komandnoj liniji idealna za sve te namjene i još puno toga. Prilikom instalacije Wiresharka na Windows operacijske sustave korisnik ima opciju instalirati i tshark, no na Linux operacijskim sustavima instalaciju je poželjno obaviti direktno u terminalu.

Pokretanje na Windows operacijskim sustavima jednako je kao i na Linux operacijskim sustavima – potrebno je pokrenuti cmd/terminal, pozicionirati se na mjesto instalacije tshark-a te ga pokrenuti naredbom „tshark“ koja prima veliki broj argumenata, čiji cjelokupni popis se može pronaći na Wiresharkovim mrežnim stranicama, odnosno izvoru [3]. Jedan od tih argumenata prikazuje sva mrežna sučelja nad kojim tshark može dohvaćati pakete, kako je vidljivo na sljedećoj slici:



Slika 9: tshark - prikaz mrežnih sučelja (izvor: slika autorice)

Argumentom „-D“ ispisuju se mrežna sučelja koja se mogu odabrati prilikom dohvaćanja paketa, a u ovome slučaju to će biti sučelje naziva Wi-Fi. Na sljedećoj slici prikazano je dohvaćanje paketa na tom sučelju koristeći dva argumenta:



Slika 10: tshark - dohvaćanje paketa (izvor: slika autorice)

Korištenjem argumenta „-i“ određuje se sučelje nad kojim se vrši dohvaćanje paketa, a argument „-f“ definira filter. U ovome slučaju odabran je filter koji prikazuje samo pakete UDP protokola, kojih je za vrijeme praćenja dohvaćeno ukupno 8, prije nego što je proces prekinut. Osim što tshark prikazuje dohvaćene pakete, iste sprema i u privremenu datoteku koju zatim korisnik može trajno pohraniti. U praktičnome dijelu rada bit će prikazano i funkcioniranje tsharka, odnosno njegovog python pandana – pysharka, unutar programskog koda.

4.1.1.1. Usporedba tsharka i tcpdump-a

Tcpdump još je jedan popularni izbor mrežnih inženjera no za razliku od Wiresharka ovaj je sustav za analizu mrežnog prometa u potpunosti baziran na terminalu. Osim toga, namijenjen je samo za Linux operacijske sustave te je za njih i optimiziran [8]. Glavna mu je svrha analiza mrežnog prometa te se koristi na serverima, ali i na lokalnim računalima. Iako je vrlo popularan i još uvijek korišten, činjenica je da je Wireshark, odnosno tshark u ovome slučaju, novija opcija s više mogućnosti te osim što se može koristiti u terminalu, ima i grafičko sučelje koje je mnogim korisnicima preglednije i lakše za koristiti. Važno je napomenuti kako su i tshark i tcpdump sustavi otvorenoga koda te su kao takvi potpuno besplatni te održavani od strane volontera i programera diljem svijeta. Oba su također redovno održavana što potvrđuju i stranice dokumentacije tcpdump-a koje su posljednji puta ažurirane u siječnju ove godine.

Na prvi pogled među njima nema previše razlike budući da oba sustava koriste terminal, no razlikuju se u mogućim atributima, filterima i načinu na koji gledaju pakete. Još jedna razlika među njima jest da je tshark puno jednostavnije (i uopće moguće) koristiti na Windows operacijskome sustavu, dok se tcpdump uglavnom koristi na Linux operacijskim sustavima. Na sljedećoj je slici prikazano pokrenuto dohvaćanje mrežnih paketa koristeći naredbe prema [8]:

```
klijent-1@klijent1-VirtualBox:~$ sudo tcpdump -i enp0s3 udp
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
19:17:08.633520 IP klijent1-VirtualBox.50484 > bud02s37-in-f14.1e100.net.https: UDP, length 1357
19:17:08.633568 IP klijent1-VirtualBox.50484 > bud02s37-in-f14.1e100.net.https: UDP, length 1357
19:17:08.633593 IP klijent1-VirtualBox.50484 > bud02s37-in-f14.1e100.net.https: UDP, length 257
19:17:08.701263 IP bud02s37-in-f14.1e100.net.https > klijent1-VirtualBox.50484: UDP, length 33
19:17:08.710661 IP bud02s37-in-f14.1e100.net.https > klijent1-VirtualBox.50484: UDP, length 29
19:17:08.711369 IP klijent1-VirtualBox.52399 > 192.168.0.1.domain: 7889+ [1au] PTR? 15.2.0.10.in-addr.arpa. (51)
19:17:08.723331 IP klijent1-VirtualBox.50484 > bud02s37-in-f14.1e100.net.https: UDP, length 33
19:17:08.732924 IP bud02s37-in-f14.1e100.net.https > klijent1-VirtualBox.50484: UDP, length 77
19:17:08.733243 IP klijent1-VirtualBox.50484 > bud02s37-in-f14.1e100.net.https: UDP, length 39
19:17:08.741826 IP 192.168.0.1.domain > klijent1-VirtualBox.52399: 7889 NXDomain 0/0/1 (51)
19:17:08.741956 IP bud02s37-in-f14.1e100.net.https > klijent1-VirtualBox.50484: UDP, length 29
19:17:08.742037 IP klijent1-VirtualBox.52399 > 192.168.0.1.domain: 7889+ PTR? 15.2.0.10.in-addr.arpa. (40)
19:17:08.742259 IP klijent1-VirtualBox.50484 > bud02s37-in-f14.1e100.net.https: UDP, length 32
19:17:08.780092 IP 192.168.0.1.domain > klijent1-VirtualBox.52399: 7889 NXDomain 0/0/0 (40)
19:17:08.791635 IP bud02s37-in-f14.1e100.net.https > klijent1-VirtualBox.50484: UDP, length 29
19:17:08.814799 IP klijent1-VirtualBox.40278 > 192.168.0.1.domain: 29176+ [1au] PTR? 1.0.168.192.in-addr.arpa. (53)
19:17:08.878786 IP 192.168.0.1.domain > klijent1-VirtualBox.40278: 29176 NXDomain 0/0/1 (53)
19:17:08.878879 IP klijent1-VirtualBox.40278 > 192.168.0.1.domain: 29176+ PTR? 1.0.168.192.in-addr.arpa. (42)
19:17:08.899646 IP 192.168.0.1.domain > klijent1-VirtualBox.40278: 29176 NXDomain 0/0/0 (42)
^C
19 packets captured
19 packets received by filter
0 packets dropped by kernel
```

Slika 11: tcpdump - prikaz dohvaćanja paketa (izvor: slika autorice)

Naredba je pokrenuta koristeći jednake uvjete kao i na slici 10, dakle slušanje je postavljeno na aktivno sučelje koje se definira atributom „-i„ jednako kao i u Wiresharku, no filtriranje po protokolu u ovome slučaju nije potrebno dodatno označiti već ga se može samo upisati. Ponovno je filter UDP protokol kako bi se lakše uočile razlike u ispisu. Također je važno napomenuti kako ovaj ispis nije toliko detaljan koliko bi bio da je uključen *verbose* ispis. Unatoč tome, tcpdump ispis izgleda detaljnije budući da je uključena i DNS rezolucija svake

IP adrese – drugim riječima, prikazan je i naziv svakog od poslužitelja osim samo IP adrese kao što je to u tsharku. Za razliku od tsharka, tcdump dodatnim atributom onemogućuje DNS rezoluciju u ispisu te paketi također nisu numerirani.

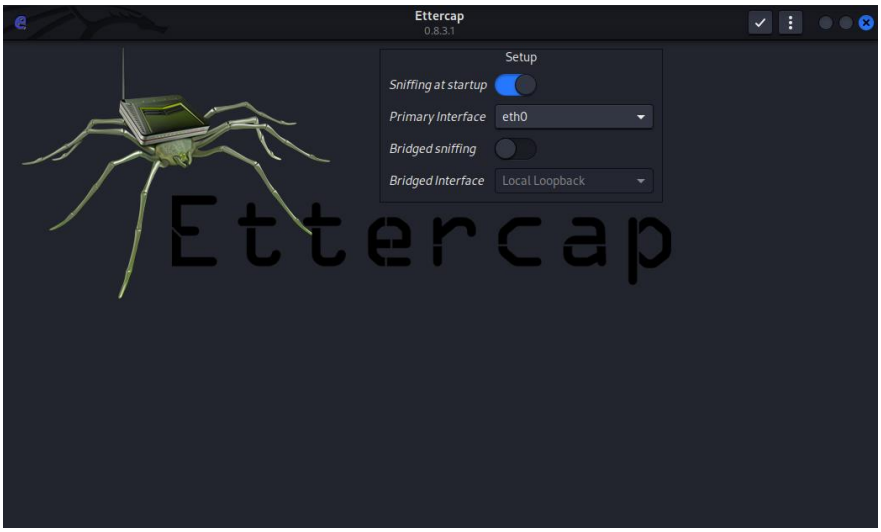
Iako na prvi pogled tcpdump ispis izgleda detaljnije od tshark ispisa, samim proučavanjem dokumentacije oba alata može se zaključiti kako tshark nudi puno više mogućnosti. Primjerice, tcpdump nema dijagnostičke mogućnosti kakve ima tshark u obliku atributa „--log-level“, „--log-fatal“ i drugih [3]. Ove će mogućnost poslužiti programerima u *debuggiranju* odnosno lakšoj dijagnostici problema u kodu. Unatoč tome, tcpdump i dalje je odabir mnogih mrežnih inženjera zbog svojeg statusa i jednostavnosti korištenja, budući da za korištenje tcpdump-a nije potrebno ništa dodatno instalirati zbog toga što isti već dođe pred-instaliran na Ubuntu, ali i mnogim drugim Linux distribucijama.

4.2. Alternativni sustavi za analizu mrežnog prometa

U ovome će poglavlju biti opisani neki sustavi za analizu mrežnog prometa koji djeluju kao alternativna Wiresharku. Capsa Portable Network Analyzer i Ettercap tako su sustavi za analizu mrežnog prometa koji u nekim svojim funkcionalnostima mogu zamijeniti Wireshark, ali ga i dodatnim mogućnostima nadopuniti. Ettercap je primjerice pripremljen za izvođenje napada na loklanu mrežu te je kao Wireshark potpuno besplatan za korištenje i otvorenoga koda, dok je Capsa Portable Network Analyzer predviđen za korporacije i druge velike mreže te ga je potrebno kupiti i licencirati.

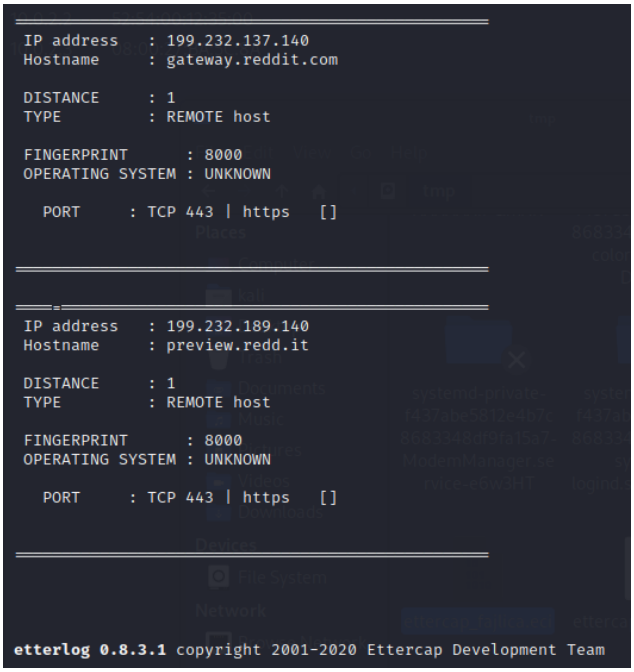
4.2.1. Ettercap

U svojim počecima Ettercap je služio isključivo kao alat za analizu mrežnog prometa – nudio je opcije dohvaćanja mrežnih paketa te je svojim funkcionalnostima bio sličniji Wiresharku [20]. Tijekom razvoja aplikacije postepeno su dodavane funkcionalnosti koje danas služe izvođenju MITM (*Man in the middle*) napada na razne načine te je *go-to* alat mnogim mrežnim inženjerima za simulaciju napada na vlastitu mrežu kako bi otkrili sigurnosne propuste i testirali postojeće sigurnosne mjere. Ettercap i dalje može poslužiti kao obični *sniffer* paketa no to mu nije primarna namjena te se i dalje razvija kao napadački alat više nego analitički kao što je to Wireshark. Ettercap je dostupan za besplatno preuzimanje (alat je otvorenoga koda) na svim popularnim operacijskim sustavima – Windows, MacOS te Linux distribucijama. Na sljedećoj slici prikazano je korisničko sučelje aplikacije pokrenuto na Kali Linux distribuciji:



Slika 12: Ettercap početno sučelje (izvor: slika autorice)

Na početnom su zaslonu prikazane opcije za biranje mrežnog sučelja nad kojim će se izvršavati analiza mrežnih paketa ili putem kojeg će se izvršavati MITM napad, zatim je dostupna opcija „Bridged sniffing“ koja omogućava „nečujni“ *sniffing* paketa na mreži putem *bridged* adaptera što znači da Ettercap koristi dva mrežna sučelja i prosljeđuje mrežne pakete s jednog na drugi dok u isto vrijeme provodi filtriranje dospjelih paketa [20]. Prihvaćanjem odabranih opcija pokreće se dohvaćanje mrežnih paketa koje je moguće pohraniti u datoteku te analizirati korištenjem pomoćne aplikacije „Etterlog“ unutar terminala. Ispis zaprimljenih paketa izgleda kako slijedi:



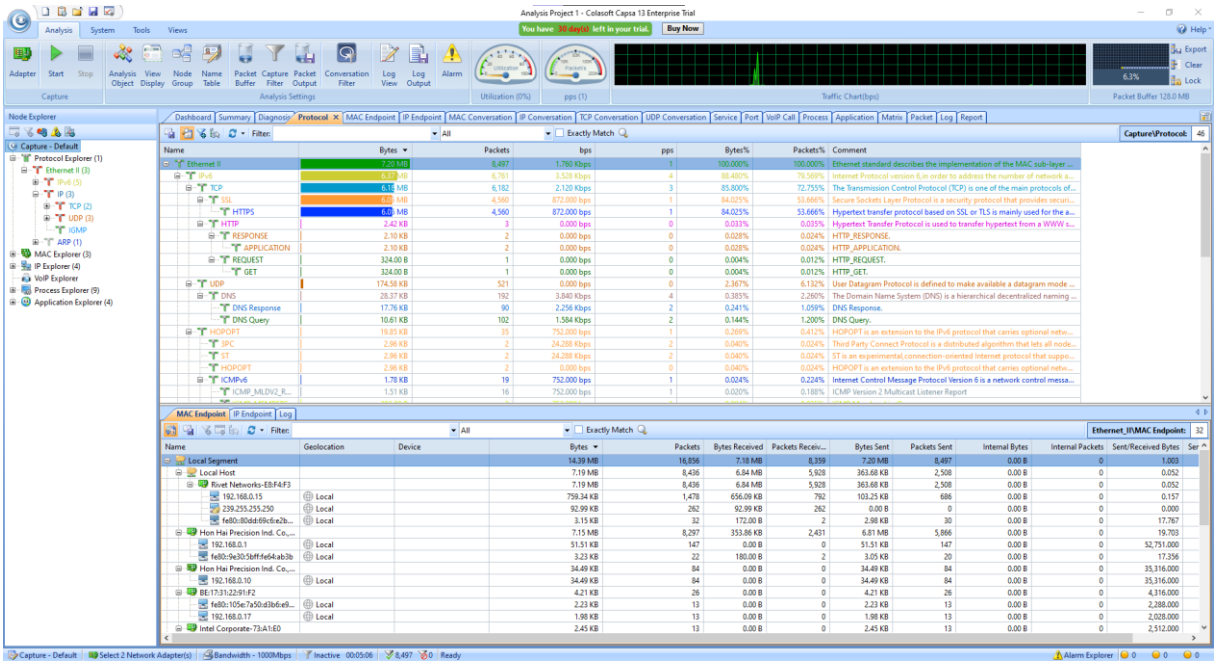
Slika 13: Prikaz dohvaćenih paketa u Ettercapu korištenjem Etterloga (izvor: slika autorice)

Na slici je uočljivo kako su, za razliku od Wiresharka, paketi opisani s puno manje detalja te ih nije moguće vidjeti jednostavnim ispisom već korištenjem raznih atributa koji omogućavaju filtriranje, detaljniji ispis, itd.

Unatoč tome, Ettercap je vrlo koristan alat prepun funkcionalnosti koje mogu poslužiti u ofenzivne, ali i defanzivne svrhe za mrežne inženjere koji žele pronaći nesigurnosti u mreži kako bi ih mogli ispraviti. Nešto kasnije u radu ovaj će alat biti korišten za izvođenje spomenutog MITM napada.

4.2.2. Capsa Portable Network Analyzer

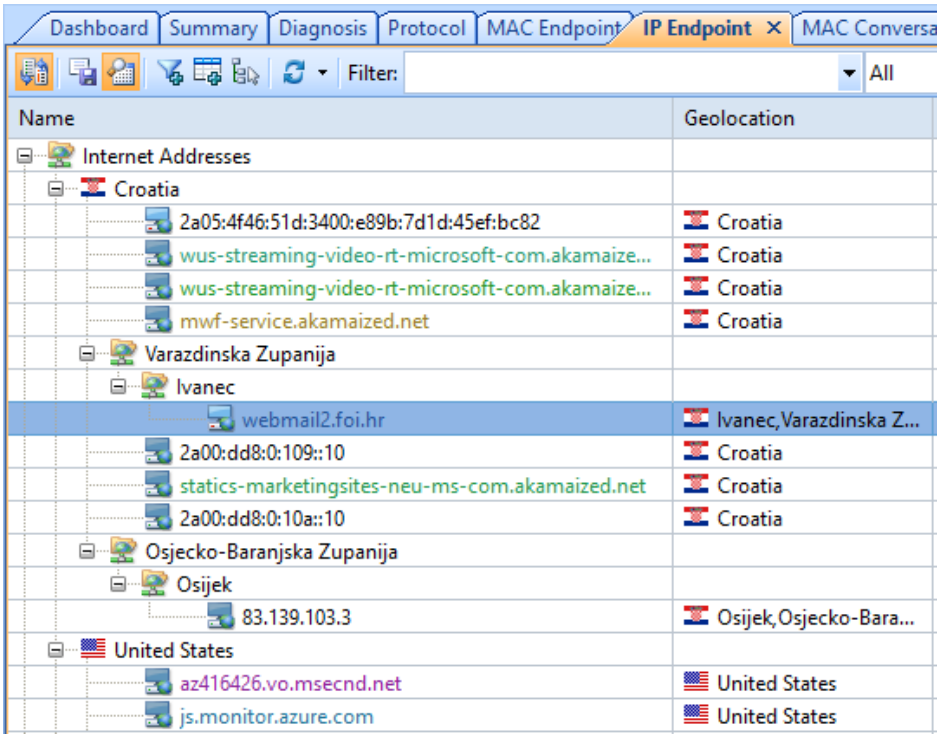
Za razliku od Wiresharka te iznad opisanog Ettercapa, Colasoftov Capsa Portable Network Analyzer je „Enterprise“ rješenje, odnosno sustav koji je namijenjen korištenju u velikim poduzećima. Capsa također nije besplatna kao Wireshark ili Ettercap te je licencu za korištenje potrebno kupiti – jednokratna cijena za korištenje na jednom uređaju jest 1295 američkih dolara, a za korištenje na pet uređaja je 5850 američkih dolara [21] te je proizvod moguće kupiti na službenoj stranici Colasofta. Za potrebe testiranja iskorišten je besplatni probni period od mjesec dana. Ovo se možda čini kao velika cijena za posao koji može obaviti i besplatni Wireshark, no pokretanjem mrežne analize postane u ovome *softveru* postaje očigledno što se zapravo naplaćuje:



Slika 14: Capsa korisničko sučelje (izvor: slika autorice)

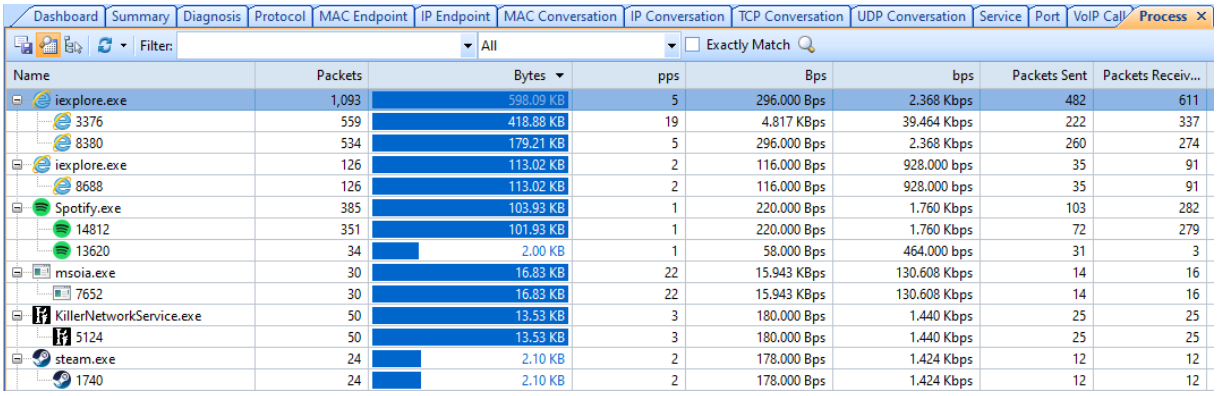
Capsu odlikuje iznimno pregledno korisničko sučelje u kojem je mrežni promet moguće analizirati kroz predefinirane filtere portova, protokola, IP adresa i svega drugoga što bi korisniku moglo biti korisno. Na slici iznad je primjerice prikazana prevalentnost protokola u ukupnome broju zaprimljenih paketa koji se zatim granaju po nižim slojevima zajedno s njihovim opisima, komentarima te podatkovnom veličinom također uzimajući u obzir ukupni broj zaprimljenih paketa. Svaki je ovaj paket moguće analizirati do u najsitnije detalje što ovaj alat čini zaista impresivnim.

Još jedna zanimljiva funkcionalnost jest prikazivanje IP adresa s kojih su paketi zaprimljeni kroz geolokaciju zemalja odakle potječu. Na sljedećoj slici raščlanjene su neke IP adrese od kojih i IP adresa FOI webmaila:



Slika 15: Capsa geolokacijski prikaz IP adresa (izvor: slika autorice)

Vidljivo je dakle kako se poslužitelj za FOI-jev webmail nalazi u Varaždinskoj županiji. Naravno, kao i u svakoj IP geolokaciji preciznost je izvediva samo do neke točke te je tako ovdje napisano kako se poslužitelj nalazi u Ivancu umjesto u Varaždinu. Na sljedećoj slici prikazana je još jedna korisna funkcionalnost:



Slika 16: Capsa prikaz aktivnih procesa (izvor: slika autorice)

Moguće je vidjeti koji je proces sudjelovao u kojem broju paketa te njihova ukupna veličina – u ovome slučaju to su Internet Explorer (na kojemu je bio otvoren Colasoftov priručnik za Capsu), a vidljivi su još od poznatih aplikacija Spotify te Steam koji su također u trenutku prikupljanja podataka bili upaljeni. Ova je funkcionalnost posebno korisna kada se želi vidjeti postoji li na računalu virus ili slični softver koji komunicira s nekim poslužiteljem te je osim

toga moguće detaljno vidjeti koji servis troši koliku količinu podataka ukoliko se radi o mreži s ograničenom potrošnjom.

Daljnjim istraživanjem Capse otkriva se veliki broj mogućnosti od kojih su vrijedne spomena još i kreiranje grafikona za IP i MAC komunikacije te automatsko kreiranje izvještaja i statistika pri završetku dohvaćanja paketa. Dohvaćene mrežne pakete također je moguće spremiti te ponovno otvoriti po potrebi, slično kao i u Wiresharku te Ettercapu no za razliku od Ettercapa nije potreban drugi alat niti se podaci ispisuju u terminalu već se otvaraju direktno na korisničkom sučelju sa svim prethodno opisanim mogućnostima. Capsa Portable Network Analyzer iznimno je koristan i impresivan alat koji svojim mogućnostima prestiže i popularni Wireshark, no cijena ga ne čini dostupnim svima već ga ima smisla koristiti samo u korporativnom okruženju.

5. Korištenje sustava za analizu mrežnog prometa

Kroz prethodna poglavlja opisana je teorijska podloga sustavima za analizu mrežnog prometa te su prikazani najčešće korišteni sustavi i njihove funkcionalnosti. U ovome poglavlju riječi će biti o njihovoj praktičnoj upotrebi te će se izvesti eksperimenti korištenja nesigurnih protokola te napad na računala u lokalnoj mreži. Bit će također opisano i korištenje ovakvih sustava u programiranju odnosno u novim aplikacijama kao dio njihovih funkcionalnosti. Kroz postavljena virtualna računala prikazat će se simulirana lokalna mreža te na koje se sve načine neispravne ili zastarjele instalacije mogu iskoristiti protiv korisnika te će naglasak biti na izbjegavanju tako opisanih nesigurnih metoda i načina postavljanja programa.

5.1. Iskorištavanje mana u mrežnim protokolima pomoću Wiresharka

Wireshark kao takav nije namijenjen agresivnom napadanju ili prisluškivanju mreže. Njegova je namjena analiza postojeće mreže ili mrežnog mjesta kako bi se izbjegli potencijalni napadi nekim drugima alatima (primjerice Ettercap). U ovome će se eksperimentu prikazati korištenje starijih protokola kao što su HTTP, SMTP, POP, FTP i IMAP te istraživanje i iskorištavanje njihovih ranjivosti. Prema [9] ovi protokoli (koji se i danas koriste na nekim mrežnim stranicama) sadrže ranjivosti koje se mogu iskoristiti u svrhu otkrivanja korisničkog imena i lozinke, razmijenjenih poruka te datoteka koje se prenose ovim protokolima. U sljedećih će se nekoliko poglavlja tako prikazati pokušaj pronalaska korisničkih podataka i datoteka nakon čega će se opisati alternativni protokoli ili sigurnije korištenje istih kako bi se poboljšala sigurnost mrežnih mjesta.

5.1.1. Testno okruženje

Za potrebe testiranja u virtualnom je okruženju postavljen Ubuntu server verzije 22.04 što je u vrijeme pisanja ovog rada najnovija dostupna verzija ovog operacijskog sustava. Na poslužiteljsko su računalo postavljeni sljedeći servisi:

- Apache server 2.4.52
- MariaDB 10.6.7
- PHP 8.1.2

čime ovaj server čini potpuni LAMP *stack.* Ideja iza postavljenih servisa jest uspostavljanje mail poslužitelja na kojemu se mogu testirati HTTP, POP3/IMAP i FTP protokoli te uočavanje njihovih nesigurnosti. Apache server i PHP u ovome će slučaju poslužiti kao servisi iza „Roundcube“ aplikacije za zaprimanje mailova, dok će MariraDB biti baza u koju će se spremati poslani i zaprimljeni mailovi. Također je važno napomenuti i kako će za potrebe testiranja biti izostavljeno postavljanje sigurnosnih mjera kao što su SSL/TLS odnosno *Secure Socket Layer/Transport Layer Security* čija je glavna svrha sigurni prijenos datoteka između protokola koristeći se metodama kao što su autentikacija, enkripcija i dekripcija čim se potencijalnim napadačima uvelike otežava krađa podataka [1] te se takva konfiguracija ne preporuča u drugim slučajevima. SSL i TLS gotovo su jednaki po funkcionalnosti te se za sigurni sloj koristi naziv SSL/TLS budući da će u svakom slučaju glavnu ulogu u sigurnosti imati jedan od ova dva protokola.

Za funkcionalni mail poslužitelj potrebno je postaviti i sljedeće servise:

- Postfix 3.6.4
- Dovecot 2.3.16
- Roundcube 1.5.3

Postfix služi za elektronički prijenos mailova, a Dovecot za čitanje i slanje mailova koristeći protokole kao što su IMAP i POP3 [11]. Zajedno ova dva servisa čine potpuni mail server, dok Roundcube aplikacija služi kao mail klijent na klijentskom računalu. Mail poslužitelj ne može biti svrha sam sebi pa je iz tog razloga postavljeno još jedno virtualno računalo na kojemu je instaliran operacijski sustav Kali Linux verzije 2022.2. Oba virtualna računala postavljena su na NAT mrežu odnosno dodijeljene su im lokalne IP adrese u obliku 10.x.x.x te mogu komunicirati kao lokalna računala no imaju i izlaz na Internet [1].

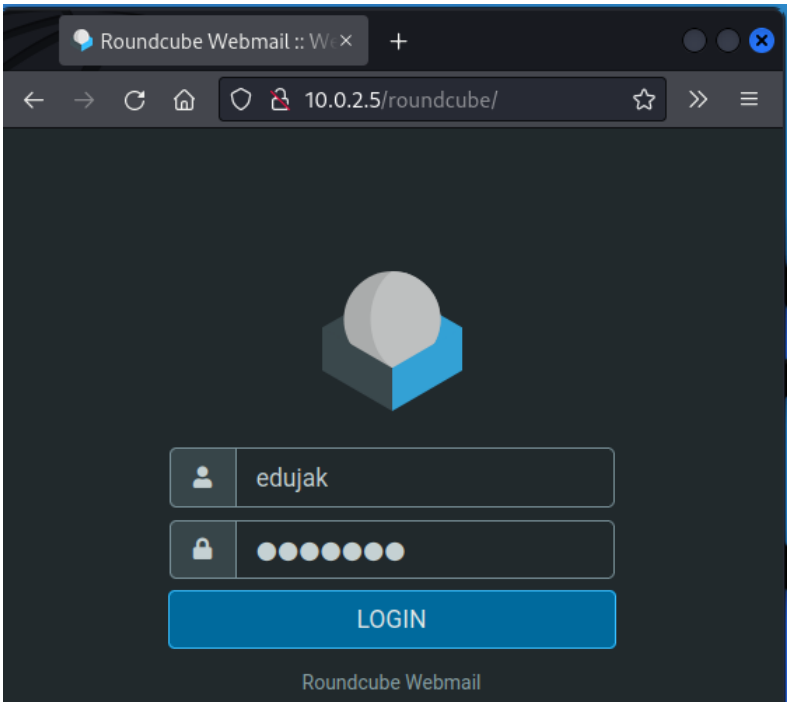
Prilikom instalacije *Postfix* servisa potrebno je pratiti upute na ekranu te ga konfigurirati po želji. Produkcijska okolina zahtijeva korištenje TLS-a no u ovome slučaju to nije potrebno pa se *Postfix* ostavlja kako je u svojoj osnovnoj verziji. *Dovecot* zahtijeva nešto više razumijevanja i konfiguracije budući da se ovdje konfigurira i mail domena te dodaju SSL certifikati [12]. U ovome slučaju neće biti korišteni certifikati niti privatni ključ no iste je u konfiguraciji moguće (i poželjno je) dodati. Na sljedećoj slici prikazan je uspješno poslan testni mail:

```
Return-Path: <edujak@foi.hr>
X-Original-To: edujak@diplomski-server
Delivered-To: edujak@diplomski-server
Received: from localhost (localhost [127.0.0.1])
        by diplomski-server (Postfix) with SMTP id D26DA77B6
        for <edujak@diplomski-server>; Sat,  6 Aug 2022 19:40:33 +0000 (UTC)
Message-Id: <20220806194040.D26DA77B6@diplomski-server>
Date: Sat,  6 Aug 2022 19:40:33 +0000 (UTC)
From: edujak@foi.hr

Bokic!
```

Slika 17: Funkcionalni mail poslužitelj (izvor: slika autorice)

Posljednji korak instalacija je i konfiguracija korisničkog sučelja odnosno mail klijenta. *Roundcube* je mail klijent otvorenoga koda kojeg svatko može instalirati i koristiti na svojem mail poslužitelju te se kao takav pokazao kao odlično rješenje za analizu mrežnog prometa. Nadalje, *Roundcube* koristi HTML formu koja POST metodom šalje podatke poslužitelju na provjeru. Na slici ispod prikazan je ekran za prijavu koji ujedno označava i uspješno konfigurirani mail poslužitelj. Budući da se radi o lokalnom poslužitelju te za adresu nije dodatno postavljen DNS moguće mu je pristupiti putem IP adrese te naziva servisa.



Slika 18: Roundcube ekran za prijavu (izvor: slika autorice)

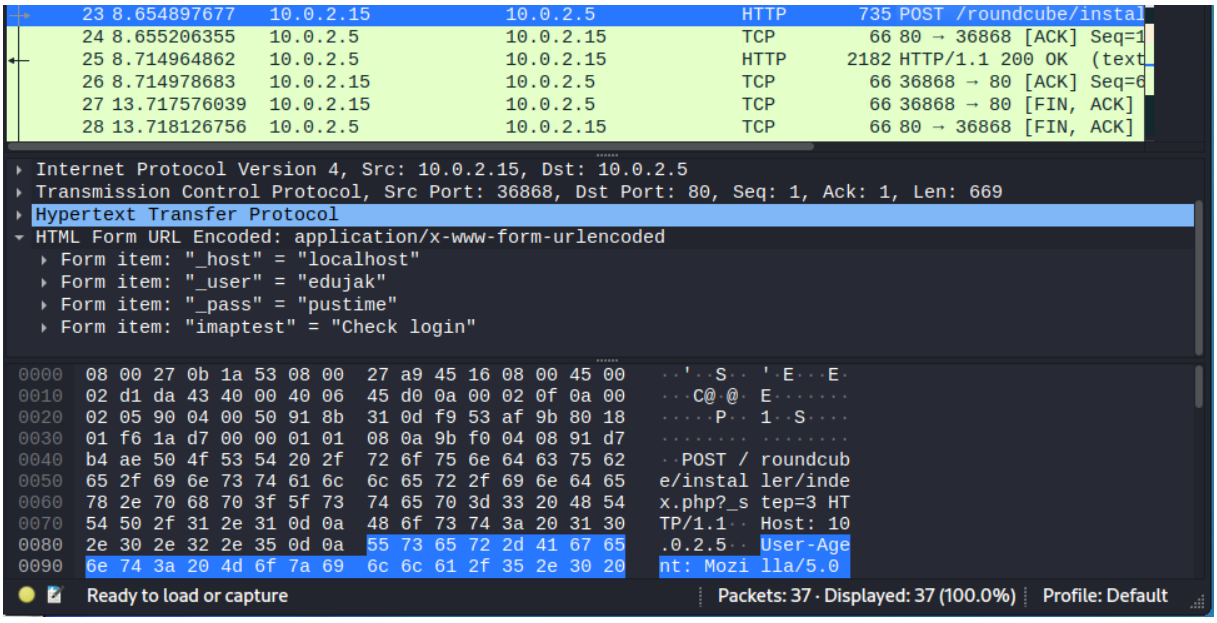
U sljedećem poglavlju bit će prikazana prijava pomoću forme na slici iznad koristeći nesigurni HTTP protokol te autentikacija pri slanju mailova koristeći protokol IMAP.

5.1.2. HTTP protokol

HTTP odnosno *Hypertext Transfer Protocol* na internetu se koristi još od 1990.-ih godina kada postaje *de facto* standard za povezivanje stranica na internetu koristeći

hiperveze [10]. Korištenje hiperveza omogućilo je nagli rast i razvoj interneta kako bi postao ono čime se svakodnevno danas služimo. Osim toga, HTTP se koristi i kao generički protokol između korisnika, proxyja itd. te drugih internetskih protokola čime se omogućuje jednostavan pristup internetskim resursima [10].

Ipak, kao što je danas općepoznato na internetu, veliki broj korisnika neke tehnologije znači i veliki broj ljudi koji iste te korisnike žele iskoristiti. Iz tog razloga nije trebalo dugo vremena da se pronađu prve slabosti HTTP-a od kojih je najveća upravo manjak enkripcije. Budući da se u počecima interneta nije toliko pazilo na sigurnost (zbog malog broja korisnika koji ga je koristio i za koje je bio namijenjen) ni HTTP nije bio zaštićen te je bio podložan napadačima. Adventom interneta tako se počinje koristiti HTTP-ova sigurnija verzija: HTTPS odnosno sigurni HTTP [1]. Naime, HTTP protokol koji se ovdje koristi nepromijenjen je te služi istoj svrsi kao i prije no sada je između HTTP-a i drugih protokola postavljen sigurni sloj SSL/TLS [1]. Uz sve iznad navedeno na internetu je i dalje moguće pronaći stranice koje ne koriste sigurni HTTP već njegovu stariju i nesigurnu verziju. Ne preporuča se pristup takvim stranicama, a pogotovo se ne preporuča upisivanje osjetljivih podataka zbog vrlo lakog načina na koji napadači mogu doći do korisnikovih podataka. Naime, podatke je moguće otkriti koristeći Wireshark, kako je prikazano na sljedećoj slici:



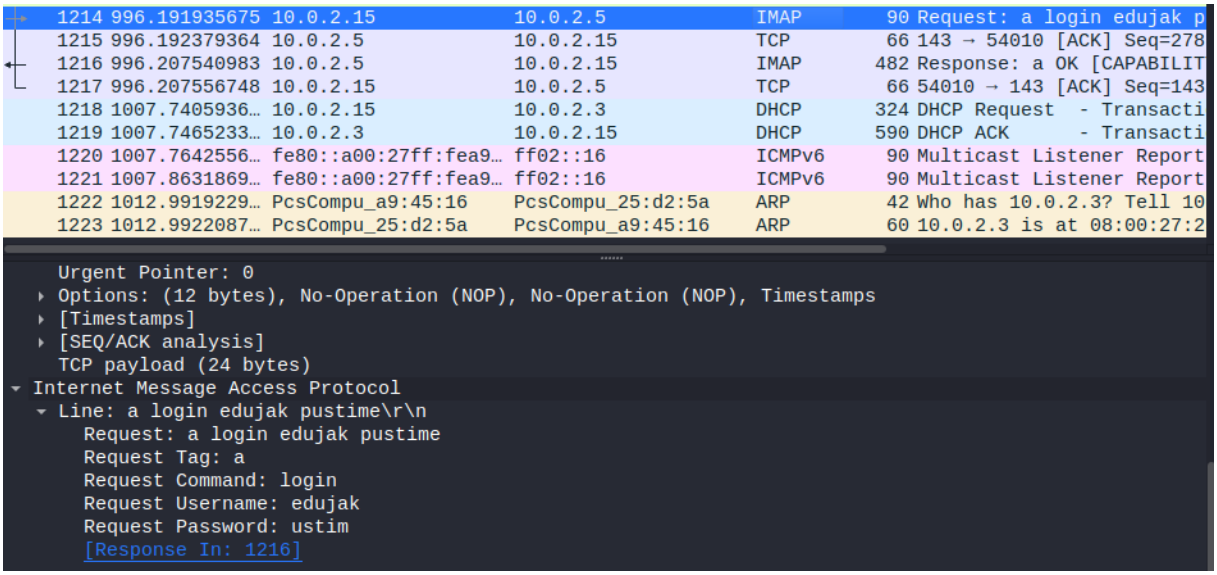
Slika 19: Prikaz prikupljenih podataka u Wiresharku – HTTP protokol (izvor: slika autorice)

Budući da se radi o nesigurnom HTTP-u koji dolazi bez SSL/TLS-a je vrlo je lako moguće doći do upisanih podataka – u ovome slučaju korisničkog imena „edujak“ i lozinke „pustime“ koji su poslani kroz formu koristeći POST metodu. Moguće je također vidjeti i podatke o kojem se poslužitelju radi te da se radi o testiranju prijave unutar GUI konfiguracije *Roundcube* servisa. Iz ovoga je vrlo lako moguće zaključiti koliko je jednostavno napadačima

doći do željenih podataka u slučaju da korisnici koriste forme na nesigurnim stranicama. U sljedećem poglavlju bit će opisan sljedeći korak u testiranju nesigurnog mail poslužitelja.

5.1.3. IMAP protokol

Korištenje e-mailova kao načina komunikacije danas je vrlo popularno i prošireno pa je tako prijeko potrebno da proces kao takav bude potpuno siguran, ali i brz. Iz tog razloga protokoli su koji se koriste u razmjeni mailova osigurani SSL/TLS-om te su i korisnički podaci i mailovi osigurani. Kako bi korisnik pristupio svojem e-mail pretincu prvo će se ulogirati u neku od aplikacija kao što je Outlook – prilikom prijave u pretinac na korisnikovo računalo korištenjem IMAP (Internet Message Access Protocol) ili POP (Post Office Protocol) protokola automatski će se preuzeti novopridošli mailovi, a za samo slanje e-mailova koristit će se SMTP (Simple Mail Transfer Protocol) [1]. Svi ovi protokoli trebali bi biti zaštićeni TLS/SSL-om no na sljedećoj slici prikazan je scenarij u kojemu je izostavljeno njihovo korištenje.



Slika 20: Prikaz prikupljenih podataka u Wiresharku - IMAP protokol (izvor: slika autorice)

U „IMAP request“ zaglavlju vidljivo je kako se radi upravo o IMAP protokolu koji poslan korištenjem telnet protokola s lokalnog računala na poslužitelja. Naime, na lokalnom virtualnom računalu pokrenuta je sljedeća naredba:

```
telnet 10.0.2.5 143
```

čime se klijentsko računalo spaja na poslužiteljsko računalo putem direktne IP adrese te definiranog porta – u ovome slučaju to je port 143 kojeg koristi IMAP i to njegova nesigurna verzija odnosno verzija bez SSL/TLS-a [12]. Nakon poslanog zahtjeva za povezivanje potrebno je upisati korisničke podatke:

```
a login edujak pustime
```

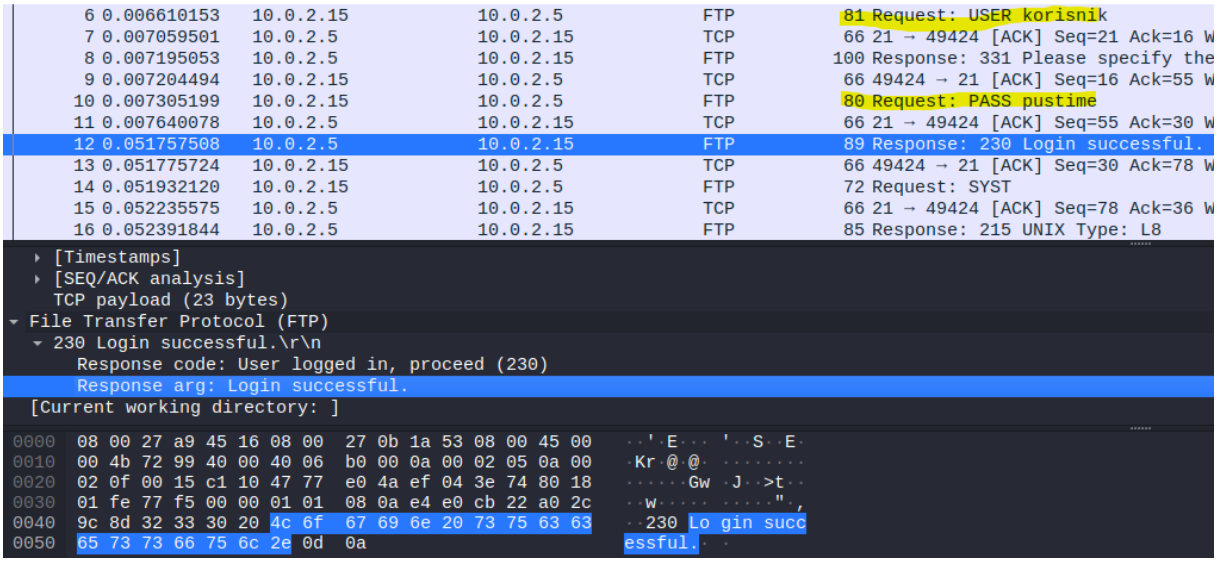
Ovom se naredbom inicijalizira prijava na poslužitelja korištenjem korisničkog imena i lozinke; budući da se radi o portu zaduženom za IMAP protokol klijentsko će računalo poslati zahtjev za dohvaćanje mailova čim se šalju i korisnički podaci. Budući da se radi o nesigurnom IMAP-u ti su korisnički podaci vidljivi te ih je moguće iščitati u Wiresharku.

Temeljem dobivenih rezultata ponovno se treba naglasiti kako je implementacija sigurnosnih mjera iznimno važna na mail poslužiteljima (ali i poslužiteljima drugih vrsta) budući da inače podaci od računala do računala putuju potpuno čitki svim napadačima koji ih pokušaju pročitati. Veliki broj korisnika mail servisa znači i veliki broj potencijalnih prijetnji te potencijalnih napadača stoga je poželjno zaštiti se na najbolji mogući način.

5.1.4. FTP protokol

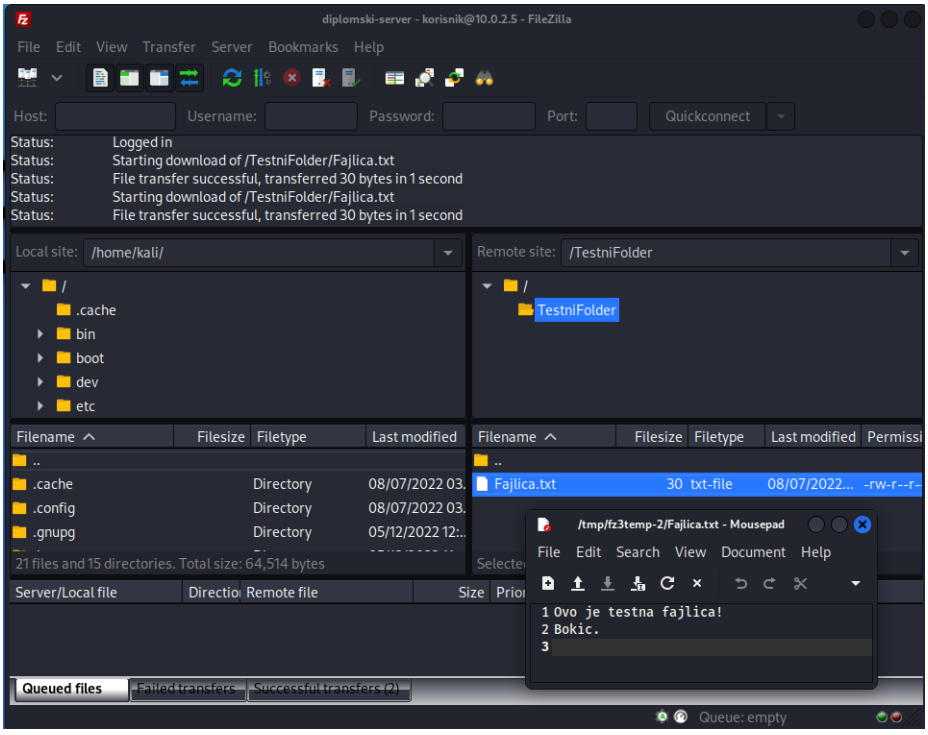
Slično kao HTTP, FTP (File Transfer Protocol) je nesigurni protokol koji je i dalje u širokoj upotrebi. U svojoj suštini radi se o jednom od osnovnih protokola koji služe prijenosu datoteka s računala na računalo te je izgrađen na temelju TCP-a, a svoju upotrebu danas pronalazi na mnogim poslužiteljima za dijeljenje datoteka te je kao takav vrlo podložan napadačima [1]. Koristi *clear-text* prijenos kredencijala za prijavu te je osim toga vrlo lako moguće pročitati i prenesene datoteke kako će i biti prikazano kasnije u poglavlju [13]. Ipak, kao i u slučaju HTTP-a, FTP također ima sigurnu verziju koja se naziva FTPS te koristi SSL/TLS za dodavanje sigurnosti [13]. Ipak, dobar dio FTP prijenosa podataka i dalje je nesiguran te će u ovome poglavlju biti prikazano korištenje jednog takvog nesigurnog poslužitelja za dijeljenje podataka te uhvaćeni podaci korištenjem Wiresharka.

Na prethodno spomenutom Ubuntu Server poslužitelju ovoga je puta instaliran i konfiguriran „vsftpd“ servis koji uspostavlja poslužitelja za usluživanje klijenata te učitavanje i preuzimanje datoteka. Ovaj servis ne nudi nikakve mogućnosti dodavanja SSL/TLS certifikata niti posebne autentikacije tijekom instalacije stoga *out of the box* dolazi kao nesigurni servis. Na *root* direktoriju kreirana je nova putanja u koju je učitana tekstualna datoteka s nekoliko rečenica sadržaja te su novokreiranom korisniku dodijeljena sva prava na toj putanji kako bi je korištenjem neke od aplikacija za upravljanje datoteka kao što je „Filezilla“ taj korisnički račun mogao ažurirati. Sljedeći korak instalacija je upravo spomenute „Filezilla“ aplikacije na klijentskom računalu te je upaljen Wireshark kako bi hvatao sav mrežni promet na aktivnom adapteru. Na sljedećoj slici prikazani su podaci dohvaćeni prijavom u poslužiteljsko računalo korištenjem navedene aplikacije:



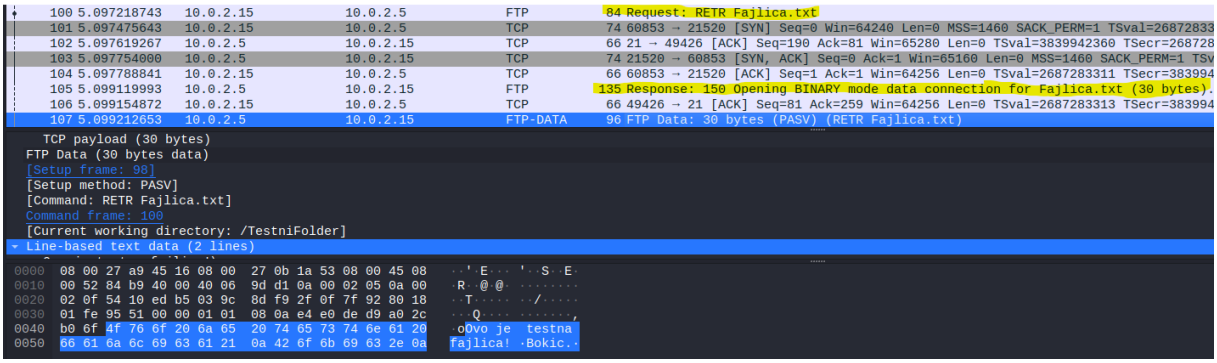
Slika 21: Prikaz prikupljenih podataka u Wiresharku - FTP protokol – kor. ime i lozinka (izvor: slika autorice)

Kao što je vidljivo, u retku 6 Wireshark je dohvatio zahtjev gdje je u zaglavlju upisano korisničko ime korisnika koji se pokušava prijaviti na poslužiteljsko računalo te je u ovome slučaju to „korisnik“. Odgovor u retku 8 traži i korisničko ime koje se šalje u retku 10 te su u zaglavlju ponovno vidljivi podaci odnosno podatak: „pustime“. Retkom 12 potvrđuje se prijava u poslužiteljsko računalo te se na „Filezilli“ ispisuju datoteke na putanji:



Slika 22: Filezilla - datoteke na predefiniranoj putanji (izvor: slika autorice)

Na putanji se, naime, nalazi točno jedna tekstualna datoteka s dvije rečenice teksta. Preuzimanjem datoteke na Wiresharku se prikazuju sljedeća zaglavlja:



Slika 23: Prikaz prikupljenih podataka u Wiresharku - FTP protokol - sadržaj (izvor: slika autorice)

U retku 100 vidljivo je kako „Filezilla“ poslužitelju šalje zahtjev za dohvaćanje datoteke na što poslužitelj odgovara u retku 105. U posljednjem vidljivom retku 107 protokol FTP – DATA dohvaća podatke te su podaci vidljivi i u Wiresharku.

Ovim se zaključuje sigurnosna analiza protokola koji su danas u širokoj upotrebi, a u sljedećem poglavlju najviše će riječi biti o korištenju Wiresharka u komandnoj liniji te prikaz kreiranog programa za aktivno praćenje mrežnog prometa.

5.2. Reaktivno praćenje mrežnog prometa korištenjem tsharka u pythonu

U prethodnom je poglavlju bilo opisano korištenje Wiresharka te njegovo grafičko sučelje pa će ovo poglavlje biti posvećeno njegovoj verziji u terminalu. Ipak, neće biti prikazano čisto korištenje tsharka putem terminala već njegov Python *wrapper* koji omogućava korištenje funkcionalnosti Wiresharka direktno u programima po potrebi korisnika [14]. Pyshark tako svoju upotrebu pronalazi u brojnim IoT projektima, ali i kućnim projektima za automatizaciju zadataka, pohranjivanje ili iščitavanje spremljenih .pcap datoteka i slično.

Za potrebe demonstracije pyshark odnosno tshark funkcionalnosti kreiran je mali program čija je svrha reaktivno praćenje mrežnog prometa kod rada na računalu kako bi se povećala koncentracija na zadatke kao što su učenje, programiranje i slično. Kroz grafičko sučelje pri pokretanju programa korisnik će odabrati vrijeme koje želi utrošiti na koncentrirani rad na računalu te će kreirani „NetworkNanny“ program u tom vremenu kontinuirano pratiti mrežni promet na korisnikovom računalu te mu zabraniti pristup internetu na određeno vrijeme kada uhvati paket s neke od „zabranjenih“ stranica kao što je u ovome slučaju Facebook. Korištenjem pyshark biblioteke program će tako prihvatiti svaki nadolazeći paket te analizirati poslužitelja s kojega je došao i u slučaju zabranjenog naziva *host* računala putem *bash* skripte zabraniti pristup internetu korisniku (ugasiti mrežni adapter koji se koristi te ga nakon određenog vremena ponovno upaliti). Kako bi bio potpuno funkcionalan uređeno

je i grafičko sučelje putem kojeg korisnik može odrediti trajanje vremena za fokus. Sami kod te funkcionalnost programa bit će prikazani u sljedećih nekoliko poglavlja.

5.2.1. Python skripta

Najbolje rješenje za pisanje koda u ovome se slučaju pokazao Kali Linux što je ujedno i vrlo popularni odabir stručnjacima za informacijsku sigurnost zbog svoje brzine, jednostavnog korištenja te činjenice da je baziran na Debianu što ga čini vrlo jednostavnim za snalaženje svakome tko je u nekom trenutku koristio Ubuntu (budući da je i sam baziran na Debianu). Kali Linux ipak nije predviđen za programski razvoj no kako se radi o mrežnoj sigurnosti te korištenju Wiresharka odnosno tsharka u ovome će slučaju biti dobar odabir. Unutar Kalija korišten je još jedan popularni izbor: Visual Studio Code i to ponovno zbog jednostavnosti korištenja ali i pregršti funkcionalnosti za veliki broj programskih jezika koje podržava.

Za početak je u program potrebno uključiti sve biblioteke koje će se koristiti pri razvoju. Važno je napomenuti i kako je korištena biblioteka Tkinter za kreiranje korisničkog sučelja te su sve pyshark funkcije u programu korištene u okviru Tkintera kao takvog [15]. Uzimajući to u obzir bit će uključene sljedeće biblioteke:

```
from tkinter import *
import pyshark
import socket
import subprocess
import threading
from datetime import datetime, timedelta
```

Kao što je već navedeno za kreiranje grafičkog korisničkog sučelja korištena je Tkinter biblioteka, a za njim slijede biblioteke „pyshark“, „socket“, „subprocess“ te „threading“. „Pyshark“ naravno služi analizi mrežnih paketa, a „socket“ služi za bolje razumijevanje istih te će u ovome programu služiti za razvrstavanje paketa po nazivu servera. Nadalje, „subprocess“ biblioteka služi za pokretanje također spomenute *bash* skripte koja preuzima kontrolu te korisniku isključuje internet na određeno vrijeme ne bi li ga podsjetio da treba biti koncentriran na rad. „Threading“ biblioteka služi za kreiranje dretvi budući da je potrebno paralelno obavljati nekoliko zadataka te je potrebno da program bude funkcionalan za trajanje izvođenja određenih procesa. Posljednja biblioteka „datetime“ služi isključivo određivanju trajanja izvođenja programa što će biti demonstrirano poslije.

```
root = Tk()

trajanje = 0
vrijeme = datetime.now()

root.title("NetworkNanny")

e= Entry(root)
e.grid(row=0, column=0)

def set_trajanje():
    global trajanje
    global vrijeme
    trajanje = e.get()
    vrijeme = (datetime.now() +
timedelta(hours=float(trajanje))).strftime('%H:%M:%S')

    labela = Label(root, text=trajanje + " sati/a neometanog
ucenja")
    labela.grid(row=1, columnspan=2)
    gumb.config(state="disabled")
    gumb2.config(state="normal")

gumb = Button(root, text="Definiraj vrijeme trajanja",
command=set_trajanje)
gumb.grid(row=0, column=1)

def analiza():
    labela2.config(text="Krecem dadiljati tvoj mrezni promet!")
    gumb2.config(state="disabled")
    while datetime.now().strftime('%H:%M:%S') < vrijeme:
        #ANALIZIRAJ MREZNI PROMET
    labela2.config(text="Gotov s dadiljanjem!")

labela2 = Label(root, text="Pokreni me!")
labela2.grid(row=2, columnspan=2, pady=10)
```

```
gumb2 = Button(root, text="Pokreni program",
command=threading.Thread(target=analiza).start)
gumb2.grid(row=3, columnspan=2, pady=15)
labela3 = Label(root, text="Izradila Ena Dujak | 2022",
fg="blue")
labela3.grid(row=4, columnspan=2)

root.mainloop()
```

Na samome početku inicijaliziran je prozor u kojemu će program biti izveden nakon čega se definiraju dvije globalne varijable od kojih će prva: „trajanje“ služiti za unos trajanja izvođenja funkcije od strane korisnika, a druga: „vrijeme“ poslužiti pri određivanju vremena u budućnosti do kada će funkcija trajati. Odnosno, korisnik će unijeti određeno vrijeme koliko želi da program radi (primjerice 2 sata), te će se ta vrijednost zbrojiti s trenutačnim vremenom kako bi funkcija za analizu mrežnog prometa znala do kada se treba izvoditi. Nakon toga definira se okvir za unos vrijednosti te gumb za dohvaćanje te vrijednosti kroz pokretanje funkcije koja određuje vrijeme trajanja mrežne analize: „set_trajanje“ u dvije prethodno opisane globalne varijable. Nakon toga korisnik treba pritisnuti drugi gumb koji pokreće analizu mrežnog prometa – ovdje se korisniku ispisuje prikladna poruka te se onemogućava prvi gumb do završavanja izvođenja funkcije. Ovaj gumb kao akciju ima kreiranje nove dretve u kojoj se izvršava funkcija analize mrežnog prometa – a ova je funkcija u „while“ petlji budući da ju je potrebno kontinuirano izvoditi dok se ne ispuni uvjet za prekidanje funkcije; u ovome slučaju to je vrijeme u budućnosti do kada treba trajati izvođenje. Kada je taj uvjet ispunjen korisniku se ispisuje poruka koja ga o tome obavještava. Zadnje linije koda odnose se na konfiguraciju labela na korisničkom sučelju te gumba za pokretanje programa.

Najvažniji dio programa ipak je dio za analizu mrežnog prometa gdje se koristi pyshark a taj dio je kako slijedi:

```
iface_name = 'eth0'
capture = pyshark.LiveCapture(interface=iface_name)
try:
    for packet in capture.sniff_continuously():
        try:
            hostic = socket.gethostbyaddr(packet.ip.dst)[0]
```

```
                if datetime.now().strftime('%H:%M:%S') >=
            vrijeme:
                    gumb.config(state="normal"
                    break
                elif hostic.endswith("fbcdn.net") or
                hostic.endswith("facebook.com"):
                    print("Bio si na zabranjenoj stranici. Nema
                interneta za tebe!")
                    subprocess.call("./skriptica.sh", shell=True)
                    if datetime.now().strftime('%H:%M:%S') >=
                    vrijeme:
                        gumb.config(state="normal")
                    break
                else:
                    print(hostic)
            except socket.herror:
                pass
    except AttributeError:
        pass
```

Prije svega potrebno je definirati i u varijablu pohraniti mrežno sučelje nad kojim se vrši analiza; u ovome slučaju to je „eth0“ (prema [16]). Nakon toga definira se varijabla u koju se kontinuirano pohranjuju podaci nad zadanim sučeljem te na temelju koje se vrši analiza podataka kroz nekoliko „try-except“ blokova. Nad svakim paketom u varijabli vrši se analiza te se prije same analize izbacuju moguće greške koje nastaju kao „socket.herror“ te „AttributeError“ te do kojih dolazi kada računalo ne može prepoznati naziv poslužitelja. Osim toga, za svaki se paket dobiv naziv poslužitelja te se odmah nakon toga provjerava vrijeme do kojega se dretva treba izvršavati budući da bi se inače „While“ petlja izvršavala beskonačno ili dok ne dok ne dođe do prvog paketa sa zabranjene stranice. Naime, kada se dođe do paketa koji je u ovome slučaju poslan od strane Facebooka potrebno je prekinuti izvođenje jer bi se inače skripta za opominjanje izvršavala za svaki paket koji je došao od strane ovog poslužitelja (a paketa je uvijek vrlo veliki broj), pa se nakon prvog primljenog paketa izvršava skripta koja korisniku na kraće vrijeme oduzima pristup internetu. Nakon što se skripta izvrši i ako nije isteklo vrijeme funkcija će se nastaviti izvršavati ispočetka kako bi se izbjegla opisana situacija. U slučaju da paket nije došao od strane zabranjene stranice naziv poslužitelja ispisat će se u terminalu što, iako nije potrebno za rad aplikacije, pomaže pri praćenju izvođenja.

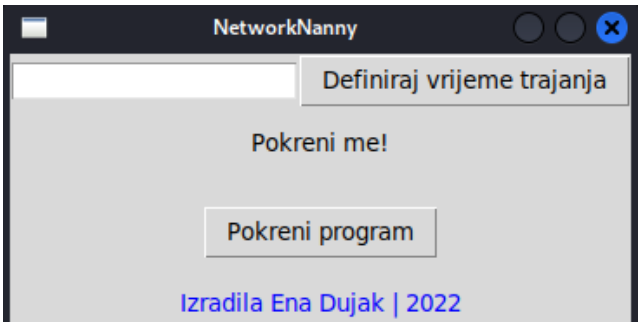
Kada korisnik dođe na stranicu na kojoj ne bi trebao trošiti vrijeme na terminalu će se ispisati poruka te će korisnik izgubiti pristup internetu na određeno vrijeme. Skripta je napisana u *bash*-u te je njen kod kako slijedi:

```
#!/bin/bash
sudo ifconfig eth0 down
echo "Nemas interneta 15 sekundi!"
sleep 15
sudo ifconfig eth0 up
echo "Opet imas internet"
```

Korištenjem „ifconfig“ naredbe prvo će se ugasiti aktivno mrežno sučelje koje će se nakon 15 sekundi ponovno upaliti – a zbog sudo naredbi koje je potrebno koristiti u skripti poželjno je da se cijeli program pokrene kao *root* korisnik odnono kroz *sudo* terminal kako se ne bi morala upisivati lozinka svaki puta kada se skripta treba izvršiti. Takav bi slučaj također bio moguć i dodjeljivanjem svih prava korisniku koji pokreće program „NetworkNanny“ no zbog sigurnosnog rizika kojeg takva akcija predstavlja preporuča se upisivati lozinku ili program pokrenuti kroz *root* okružje.

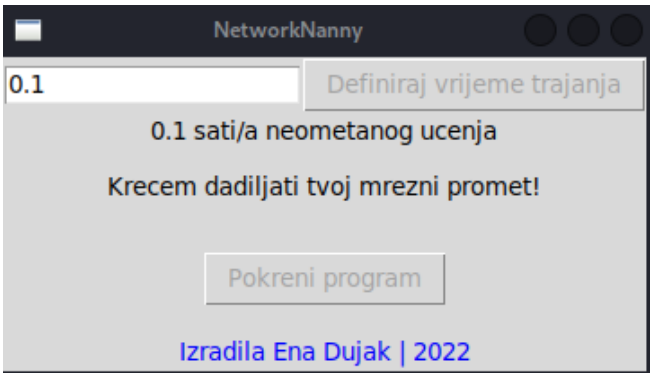
5.2.2. Pokretanje programa

Opisanim kodom moguće je prijeći na prikaz korisničkog sučelja, a ono je vrlo jednostavno i sastoji se od nekoliko osnovnih naredbi:



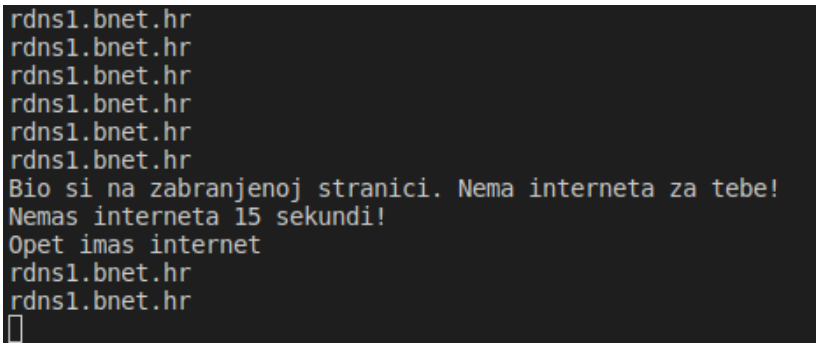
Slika 24: Prikaz GUI-ja programa (izvor: slika autorice)

Dakle, radi se o jednostavnom korisničkom sučelju koje sadrži nekoliko funkcionalnosti, a prije pokretanja njegove glavne funkcije potrebno je upisati vrijeme njegovog trajanja. U ovome slučaju to će u svrhu demonstracije biti 0.1 sat ili oko 6 minuta, a nakon toga je moguće pokrenuti program kako bi krenuo u reaktivnu analizu mrežnog prometa.



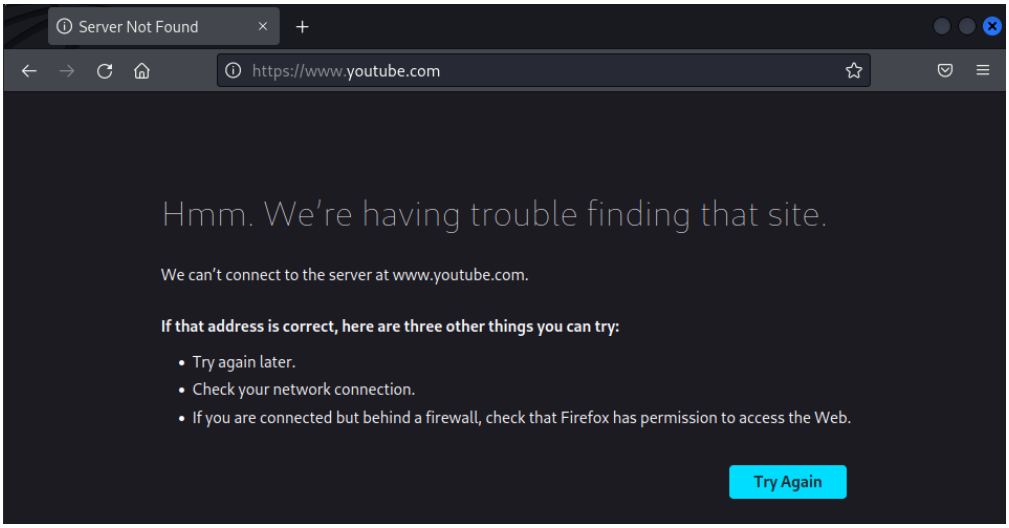
Slika 25: Pokrenuti program (izvor: slika autorice)

Korisniku se ispisuje poruka s trajanjem izvođenja programa te se onemogućava korištenje gumba kako korisnik ne bi ponovno pokrenuo dretvu za izvođenje ili prepisao vrijednost globalne varijable. Paralelno s tim i potpuno po želji korisnik tijek izvođenja može pratiti u terminalu gdje se ispisuju nazivi svih servera s kojima je računalo komuniciralo kako je i vidljivo na sljedećoj slici:



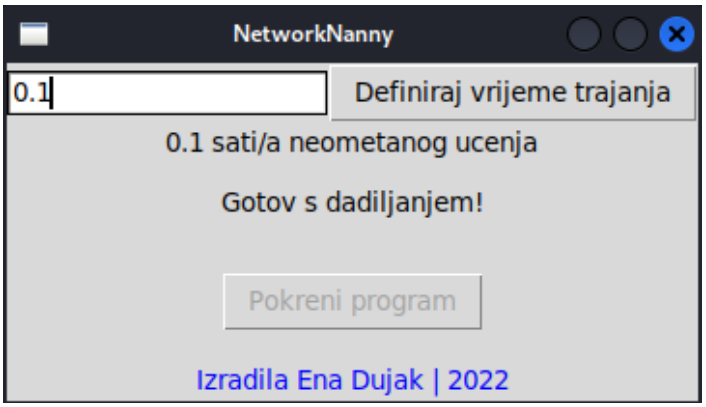
Slika 26: Ispis tijeka izvođenja aplikacije (izvor: slika autorice)

Naime, u ispisu je također vidljiva poruka koju je korisnik dobio prilikom pristupanja zabranjenoj stranici te mu je za kratko vrijeme oduzet pristup internetu:



Slika 27: Nemogućnost pristupa internetu. (izvor: slika autorice)

Nakon što je prošlo određeno vrijeme (u ovome slučaju 15 sekundi) korisniku se vraća pristup internetu te može nastaviti normalno surfati. Kako ne bi došlo do grešaka skripta će se uvijek izvršiti do kraja, čak i ako je isteklo vrijeme izvođenja funkcije kako bi korisnik po završetku izvođenja mogao nastaviti normalno surfati internetom bez briga o ponovnom pokretanju aktivnog mrežnog sučelja. Po završetku svog izvođenja funkcija će ponovno omogućiti gumb za definiranje vremena učenja te se cijeli proces može ponoviti:



Slika 28: Funkcija završila izvođenje (izvor: slika autorice)

Na slici je vidljiva poruka korisniku te korisnik sada može ponovno pokrenuti izvođenje mrežne analize ili ugasiti program.

Ovakav program samo je mali dio prikaza mogućnosti tsharka odnosno njegovog Python pandana no i dalje je vrlo koristan svima koji imaju problema pri koncentraciji u učenju ili radu na računalu. Također je važno napomenuti i da kada je velika brzina obrade paketa vitalna za rad programa treba imati na umu da je pyshark nešto sporiji od originalnog tsharka budući da koristi brojne druge biblioteke i funkcije za obradu podataka stoga se njegova upotreba u takvim slučajevima ne preporuča.

5.3. Izvođenje *Man in the middle* napada koristeći Ettercap

Kroz prethodna poglavlja prikazane su nesigurnosti u protokolima te je bilo riječi o potencijalnim napadačima i o tome što sve napadači zapravo mogu otkriti kada se ne poduzmu ispravne sigurnosne mjere. Ovo će poglavlje tako biti posvećeno upravo tome te će se iz perspektive napadača vidjeti koliko je zapravo jednostavno napraviti *Man in the middle* napad i na takav način otkriti korisničke podatke kao i „prisluškivati“ surfanje računala kojeg se napada. Prije toga potrebno je definirati pojmove koji će biti opisani – u ovome slučaju to su *Man in the Middle (MITM)* te *ARP poisoning.*

Prvi je pojam jednostavan te se odnosi na računalni napad u kojem je napadač u topologiji mreže pozicioniran između ciljanog računala te njegovog usmjerivača ili prespojnika, a za izvršavanje takvog napada koristi se neka od brojnih metoda od kojih je

vrlo popularan odabir upravo *ARP poisoning* [18]. Kako bi se napadač pozicionirao između računala i usmjerivača iskoristit će vrlo slabu sigurnost ARP tablica te njihovom izmjenom osigurati potrebne uvjete za izvršavanje napada. ARP tablica nalazi se na svakom računalu priključenom na mrežu te sadrži zapise svih IP adresa s kojima su uparene MAC adrese [17]. Na takav način računalo u nekoj mreži ne mora slati upit o uparivanju MAC adrese s IP adresom računala kada tom računalu želi nešto poslati, već u memoriji ima zapisane potrebne podatke. Međutim, ARP je vrlo jednostavan – do te mjere da je ARP zapise vrlo lako moguće lažirati te je čak na velikom broju uređaja moguće slati ARP odgovore bez da je prije toga bio poslan zahtjev [17]. Upravo zbog te jednostavnosti te planirane efikasnosti postoji i veliki sigurnosni rizik koji se iskorištava u obliku MITM napada.

ARP poisoning proces je u kojemu napadač povezuje svoju MAC adresu s IP adresom računala koje se napada, a ujedno svoju IP adresu postavlja kao *default gateway* kako bi računalo zahtjeve prema vanjskom poslužitelju slalo kroz računalo napadača, a vanjski poslužitelj ponovno odgovore šalje napadačevom računalu koje onda kroz funkciju *ip forwardinga* dobivene pakete prosljeđuje originalnom računalu [17]. Na ovakav način na računalu napadača može biti vidljiv sav mrežni promet ciljanoga računala te ako se pristupa nesigurnim stranicama moguće je čak i doći do lozinki i drugih važnih podataka. Manjak ovoga procesa jest potreba da napadač bude u lokalnoj mreži – ARP tablicu računalo izvan mreže ne može uređivati pa je tako potrebno prvo preuzeti kontrolu nad računalom u mreži ili se na neki drugi način infiltrirati u mrežu [17].

U sljedećem primjeru bit će prikazan upravo gore opisani primjer – napadačko (virtualno) računalo pozicionirat će se između Ubuntu virtualnog računala te njegovog preusmjerivača (u ovome slučaju to će biti *host* računalo) korištenjem Ettercap besplatnog alata za analizu mrežnog prometa. Uhvaćeni promet spremit će se korištenjem tsharka te vizualizirati Wiresharkom.

5.3.1. Postavljanje virtualnih računala

Budući da se radi o ozbiljnom napadu na lokalnu mrežu važno je napomenuti da su u ovome scenariju korištena virtualna računala s izlazom na Internet, ali u zatvorenom NAT okruženju. U takvome okruženju sva virtualna računala imaju izlaz na internet no na internetu nisu vidljiva kao samostalna već kroz IP adresu *host* računala – u kreiranoj virtualnoj mreži svako od ovih računala ima svoju prevedenu lokalnu IP adresu. U mreži su postavljena tri računala:

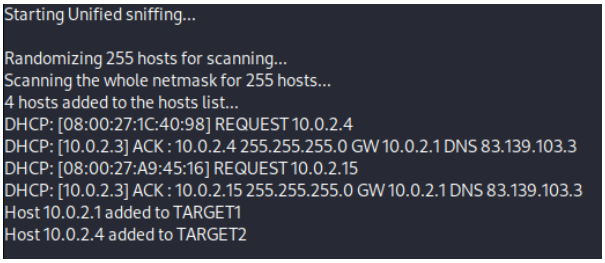
- Kali Linux (napadač) – IP: 10.0.2.15
- Ubuntu Desktop (meta) – IP: 10.0.2.4

- Ubuntu Server (poslužitelj mail servisa) – IP: 10.0.2.5

Default gateway za sva računala u mreži je 10.0.2.1, a *subnet* maska 255.255.255.0. Dakle, Kali Linux odnosno računalo napadača kao cilj će imati prisluškivanje mrežnog prometa nad Ubuntu Desktop računalom te želi saznati korisničko ime i lozinku koje ovo računalo koristi za prijavu na nesigurni lokalni mail poslužitelj (postavljanje mail poslužitelja prikazano je u poglavlju iznad). Sva računala u mreži trebaju moći međusobno komunicirati te biti vidljiva jedna drugima.

5.3.2. Izvođenje napada i analiza rezultata

Ettercap kao sustav za analizu mrežnog prometa, ali i alat za simuliranje i izvođenje raznih napada na mrežu, dolazi već instaliran na Kali Linux operacijskom sustavu te ga je za korištenje dovoljno pokrenuti na grafičkom sučelju ili terminalu. Vrlo jednostavno grafičko sučelje Ettercapa olakšava izvođenje napada te je za početak potrebno odabrati mrežno sučelje na kojemu je napadačevo računalo aktivno (u ovome slučaju to je eth0) te pritisnuti na kvačicu čime se započinje skeniranje lokalne mreže. Nakon toga i prema [18] potrebno je odabrati mete odnosno metu ili računalo nad kojim se napad izvršava. Iz izbornika se odabire opcija „Hosts“ zatim „Hosts list“ – kao prva meta dodaje se *Default gateway* IP adresa koja je u ovome slučaju 10.0.2.1 te IP adresa ciljanog računala odnosno 10.0.2.4 te ako je sve ispravno postavljeno ispisat će se sljedeća poruka:

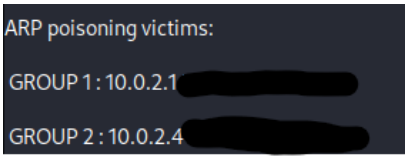


Slika 29: Prikaz meta za napad u Ettercapu (Izvor: slika autorice)

Ovako postavljeni Ettercap je sada spreman za izvršavanje napada no prije toga potrebno je još omogućiti *IP forwarding* na samom napadačkom računalu kako bi se omogućilo prosljeđivanje zaprimljenih paketa s napadnutog na napadačko računalo i dalje prema usmjerniku. Ta se opcija prema [18] omogućava sljedećom linijom u naredbenom retku:

```
sudo sysctl -w net.ipv4.ip_forward=1
```

Nakon što je sve postavljeno, na izborniku napada u Ettercapu odabire se željeni napad – u ovome slučaju to je prethodno opisani *ARP poisoning;* Ettercap sada ispisuje sljedeću poruku čime je potvrđeno da je napad pokrenut i izvršava se:



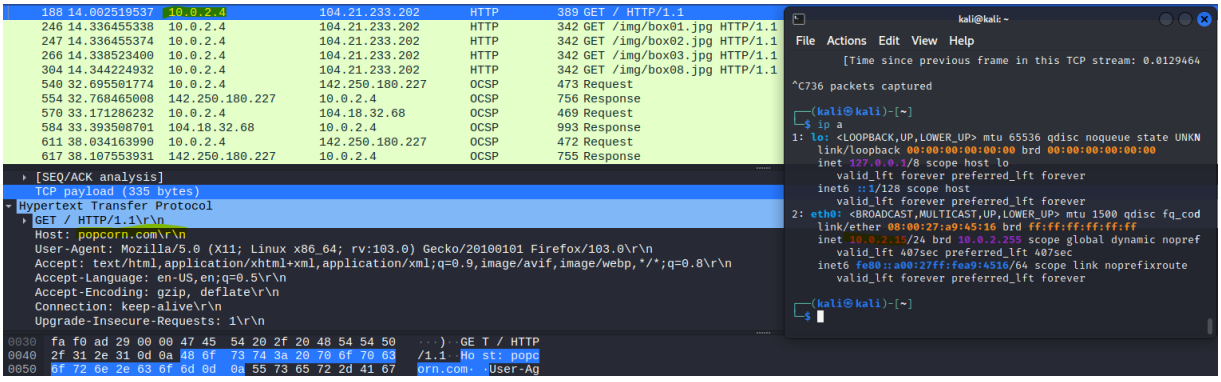
Slika 30: Pokrenuti MITM napad (Izvor: slika autorice)

Nakon što je pokrenut napad na istom je računalu u terminalu moguće pokrenuti sustav za analizu mrežnog prometa po želji, a u ovome slučaju to je bio tshark zbog jednostavnosti izvedbe i to korištenjem sljedeće naredbe:

```
tshark -i eth0 -f "port 80" -V -w ettercap_scan.pcap
```

Dakle, na mrežnom sučelju korištenjem atributa „-i“ odabire se „eth0“, zatim se određuje i filter za spremanje paketa koji je u ovome slučaju „port 80“ i to korištenjem atributa „-f“. Atribut „-V“ označava detaljnije spremanje paketa, a „-w“ atributom se definira izlazna datoteka. Na napadnutom računalu u to je vrijeme posjećena nesigurna stranica „popcorn.com“, nekoliko sigurnih stranica kao što je „reddit.com“ te nesigurni lokalni mail poslužitelj gdje je upisana lozinka i korisničko ime za prijavu. Napadač na ovakav način može doći do vrlo privatnih informacija stoga je važno poduzeti potrebne mjere predostrožnosti koje će biti opisane nešto kasnije, ali i paziti da se ne pristupa nesigurnim stranicama na internetu.

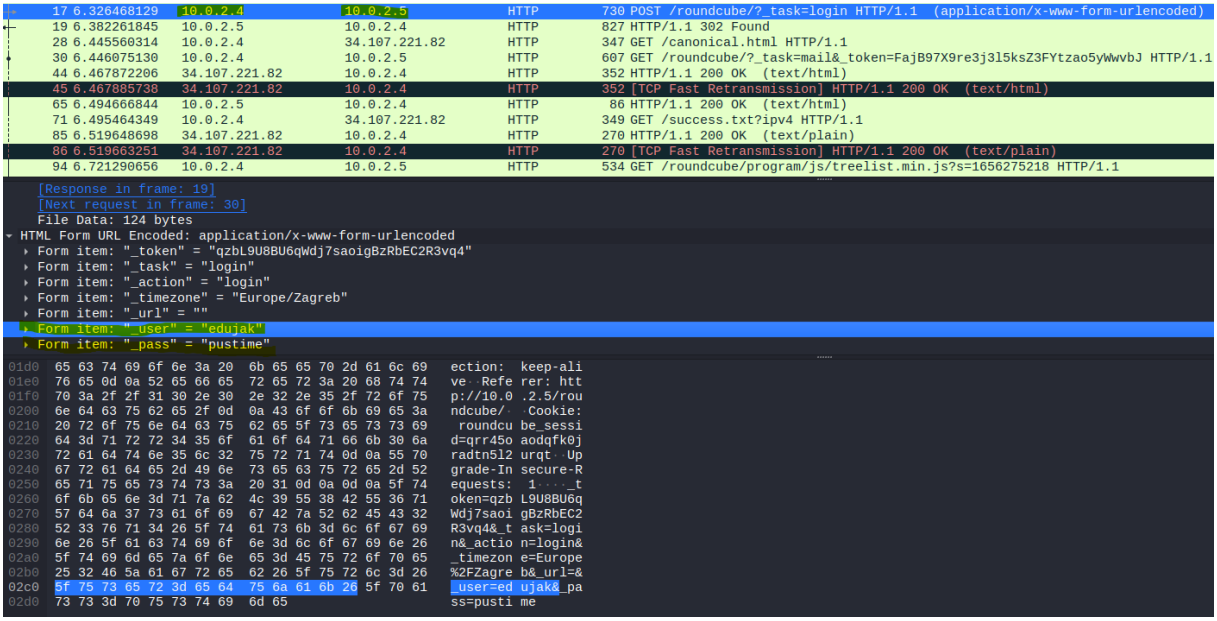
Izvođenje napada može trajati po želji no u ovome je slučaju bilo potrebno svega nekoliko minuta budući da su prikupljeni svi potrebni podaci. Ovako spremljenu datoteku moguće je otvoriti u terminalu korištenjem tsharka, ali za potrebe demonstracije datoteka će biti otvorena u Wiresharku. Na sljedećoj slici prikazan je prvi podatak od interesa:



Slika 31: Prikaz prikupljenih podataka nakon MITM napada (Izvor: slika autorice)

Prikaz terminala na desnoj strani slike pokazuje da se radi o napadačkom računalu čija je IP adresa 10.0.2.15 budući da Wireshark na lijevoj strani slike prikazuje pakete čije je izvorište

10.0.2.4 – što je adresa napadnutog računala. Iz toga je moguće zaključiti da je napad bio uspješan te sada samo preostaje analizirati uhvaćene pakete. Na slici iznad također je vidljivo da je korisnik pristupio nesigurnoj stranici „popcorn.com“ te su vidljivi svi klasični detalji kao da se radi o običnom hvatanju paketa na vlastitom računalu. Vidljivi su svi poslani zahtjevi kao i nazivi učitanih slika. Na sljedećoj slici vidljivi su i rezultati postavljenoga cilja s početka poglavlja:



Slika 32: Prikaz prikupljenih podataka nakon MITM napada - mail (Izvor: slika autorice)

Naime, u prvom retku napadnuto računalo šalje POST zahtjev prema nesigurnom mail poslužitelju za prijavu na svoj e-mail pretinac pri čemu su u paketu vidljivi svi upisani podaci. Ponovno se radi o korisničkom imenu i lozinki kao iz poglavlja o nesigurnim protokolima te je nekoliko redaka poslije vidljiva i potvrdna poruka koja označava uspješnu prijavu u mail sustav. U jednom od paketa čak je moguće vidjeti i potpuni HTML kod stranice na temelju čega napadač može točno vidjeti sve elemente na pristupljenoj web stranici. Dakle, nedvojbeno je da se radi o napadu na privatnost korisnika. Zastrašujuće je zapravo s kolikom se jednostavnošću ovakav napad može izvesti uz besplatni alat te kakve informacije se iz napada mogu izvući.

5.3.2.1. Kako se zaštititi od *Man in the middle* napada?

Po izvršavanju napada jedino što preostaje je opisati kako se od takvog napada zaštiti. Naime, sva računala s mogućnosti pristupa internetu koriste ARP tablice budući da se radi o jednom od bazičnih elemenata interneta koji je prisutan od njegovih samih začetaka. Najbolje rješenje bilo bi ARP tablice zamijeniti sigurnijim rješenjem no za sada i za računala koja ih dalje koriste postoje određene opcije koje korisnici mogu implementirati u vlastitoj mreži. Izvor [18] daje odlične prijedloge:

- Za manje mreže – statične IP adrese i statične ARP tablice rješavaju problem budući da nije potrebno kreiranje i slanje ARP zahtjeva te sva računala u mreži već znaju MAC adrese svih drugih računala.
- Za veće mreže – korištenje „Port security“ značajki kojima se onemogućava izmjena ARP tablica na jednostavan način budući da preusmjerivač u takvoj mreži dozvoljava samo jednu MAC adresu za svaki fizički port.

Ipak, svaka od ovih sigurnosnih mjera i dalje se može zaobići; u manjim se mrežama primjerice kao najveći problem predstavlja upravo njeno povećanje budući da je za svako novo računalo potrebno ručno ažurirati ARP tablice no čak se i takvi procesi isplate uzimajući u obzir vrstu i količinu podataka koje napadač jednostavnim infiltriranjem u lokalnoj mreži može presresti i ukrasti.

6. Zaključak

Tijekom pisanja ovog rada prikazala sam teoriju iza sustava za analizu mrežnog prometa te prikazala njihovu namjenu i njihove funkcionalnosti. No, puno sam pisala i o mrežnoj sigurnosti općenito te shvatila koliko je zapravo jednostavno otkriti osobne podatke unutar lokalne mreže ali i izvan nje ukoliko se koriste nesigurni protokoli i općenito sustav koji se ne nadograđuje dovoljno često.

Koristila sam tijekom svojeg istraživanja razne sustave za analizu mrežnog prometa i njima slične tehnologije ali svaki puta došla do istog zaključka – internet kao takav nije zamišljen kao sveobuhvatna mreža kakvu koristimo danas. To je vidljivo upravo u njegovim samim temeljima – ARP tablice osnova su komunikacije na internetu, a nevjerojatno ih je lagano prevariti od strane napadača koji može imati samo osnovno znanje o mrežama. Ipak, brojne nadogradnje zastarjelih protokola i procesa koji se odvijaju kako bi internet funkcionirao normalno i sigurno dokaz su tome da internet i nije ono što je bio kada je tek pokrenut. Danas je osnova svakom poslovanju, školovanju i općenito životu i kao takav treba biti potpuno siguran. Kako bi to bio, sustavi za analizu mrežnog prometa u tome uvelike pomažu te služe otkrivanju nesigurnosti u mreži koje je potrebno popraviti kako bi mreža bila što sigurnija. Koliko god je naglasak na potrebi održavanja poslužitelja i korisničkih računala sigurnima, svakodnevni korisnici interneta trebaju paziti kakvim stranicama pristupaju te kako se ponašaju u svojoj lokalnoj mreži kako bi i na taj način minimizirali mogućnost iskorištavanja slabosti.

Mogu još reći kako sam uspješno ispunila sve zadane ciljeve s početka rada te sam usput i puno toga naučila te postavila temelje za budući rad u branši mrežne sigurnosti. Mrežna je sigurnost iznimno zanimljiva i dinamična grana informatike ne samo zbog stalnih prijetnji koje su svakim danom sve kompleksnije nego i zbog tzv. „sive zone“ u kojoj se stručnjak za sigurnost mora staviti u ulogu napadača kako bi što efikasnije uvidio i potom ispravio slabosti u svojoj mreži. U pisanju sam koristila brojne stručne i znanstvene radove na temu mrežne sigurnosti te također uviđam i kako je starija literatura danas gotovo neprimjenjiva u praksi zbog stalnih nadogradnji, promjena i noviteta koji su uvedeni kako bi internet kao takav bio što sigurniji.

Na kraju pisanja ovoga rada zaključujem kako internet danas nije isto što je bio u svojim začecima te će gotovo sigurno biti potpuno drugačiji nekoliko desetljeća u budućnosti.

Popis literature

[1] L. L. Peterson i B.S. Davie, Computer networks: a systems approach, 5. izd. Burlington: Elsevier, 2012.

[2] D. S. Kim, „Application and Layered Architectures“, 2022. [Na internetu]. Dostupno na: https://et.engr.iupui.edu/~dskim/Classes/ECE547/In-network-ch02.pdf. [Pristupljeno: 27.5.2022.].

[3] R. Sharpe, E. Warnicke i U. Lamping, „Wireshark User's Guide“, 2022. [Na internetu]. Dostupno na: https://www.wireshark.org/docs/wsug_html_chunked/index.html. [Pristupljeno: 28.5.2022.]

[4] P. Mallory, „Network traffic analysis for IR: Alternatives to Wireshark“, 2019. [Na internetu]. Dostupno na: https://resources.infosecinstitute.com/topic/network-traffic-analysis-for-ir-alternatives-to-wireshark/. [Pristupljeno: 28.5.2022.]

[5] PacketLife, „CloudShark Brings Wireshark to the Web“, 2010. [Na internetu]. Dostupno na: https://packetlife.net/blog/2010/jun/22/cloudshark-brings-wireshark-web/. [Pristupljeno: 28.5.2022.]

[6] S. Petryschuk, „What is QUIC? Everything You Need to Know“, 2021. [Na internetu]. Dostupno na: https://www.auvik.com/franklyit/blog/what-is-quic-protocol/. [Pristupljeno: 29.5.2022.]

[7] M. Bradlex, „TCP vs. UDP“, 2020. [Na internetu]. Dostupno na: https://www.lifewire.com/tcp-headers-and-udp-headers-explained-817970. [Pristupljeno: 29.5.2022.]

[8] TCPDUMP&LIBCAP, „MAN PAGE OF TCPDUMP“, 2022. [Na internetu]. Dostupno na: https://www.tcpdump.org/manpages/tcpdump.1.html. [Pristupljeno: 30.5.2022.]

[9] L. Williams, „Wireshark Tutorial: Network & Passwords Sniffer“, 2022. [Na internetu]. Dostupno na: https://www.guru99.com/wireshark-passwords-sniffer.html. [Pristupljeno: 20.6.2022.]

[10] T. Berners-Lee, R. Fielding i H. Frystyk, „Hypertext transfer protocol--HTTP/1.0“, 1996. [Na internetu]. Dostupno na: https://www.rfc-editor.org/rfc/rfc1945.html. [Pristupljeno: 14.7.2022.]

[11] VULTR.COM, „How to Install Postfix, Dovecot, and Roundcube on Ubuntu 20.04“, 2020. [Na internetu]. Dostupno na: https://www.vultr.com/docs/how-to-install-postfix-dovecot-and-roundcube-on-ubuntu-20-04/. [Pristupljeno: 15.7.2022.]

[12] speedguide.net, „Port 143 Details“, 2022. [Na internetu]. Dostupno na: https://www.speedguide.net/port.php?port=143. [Pristupljeno: 1.8.2022.]
[13] N. Lord, „What is FTP Security? Securing FTP Usage“, 2018. [Na internetu]. Dostupno na: https://digitalguardian.com/blog/what-ftp-security-securing-ftp-usage. [Pristupljeno: 7.8.2022.]
[14] K. Newt, „pyshark“, 2022. [Na internetu]. Dostupno na: https://github.com/KimiNewt/pyshark. [Pristupljeno: 10.8.2022.]
[15] J. Elder, „Threading With Tkinter - Python Tkinter GUI Tutorial #97“, 2020. [Na internetu]. Dostupno na: https://www.youtube.com/watch?v=jnrCpA1xJPQ&ab_channel=Codemy.com. [Pristupljeno: 12.8.2022.]
[16] T. Galbraith, „Capturing Network Traffic With Python And TShark“, 2021. [Na internetu]. Dostupno na: https://tateg.medium.com/capturing-network-traffic-with-python-and-tshark-19599d39dbce. [Pristupljeno: 12.8.2022.]
[17] C. Nachreiner, „Anatomy of an ARP Poisoning Attack“, 2022. [Na internetu]. Dostupno na: https://csci6433.org/Papers/Anatomy%20of%20an%20ARP%20Poisoning%20Attack%20_%20WatchGuard.pdf. [Pristupljeno: 15.8.2022.]
[18] E. Gimbel, „Windows 10 ARP Spoofing with Ettercap and Wireshark“, 2020. [Na internetu]. Dostupno na: https://cybr.com/cybersecurity-fundamentals-archives/windows-10-arp-spoofing-with-ettercap-and-wireshark/. [Pristupljeno 15.8.2022.]
[19] kali.org, „What is Kali Linux?“, 2022. [Na internetu]. Dostupno na: https://www.kali.org/docs/introduction/what-is-kali-linux/. [Pristupljeno 16.8.2022.]
[20] Irongeek.com, „Manual Reference Pages - ETTERCAP (8)“, 2022. [Na internetu]. Dostupno na: https://www.irongeek.com/i.php?page=backtrack-3-man/ettercap. [Pristupljeno 17.8.2022.]
[21] Colasoft, „Capsa Portable Network Analyzer“, 2022. [Na internetu]. Dostupno na: https://www.colasoft.com/capsa/. [Pristupljeno 18.8.2022.]

Popis slika