SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 256

Analizator mrežnog protokola korištenjem Needleman Wunsch algoritma za poravnanje

Goran Peretin

Zagreb, lipanj 2008.

ZAVRŠNI ZADATAK br. 256

Zadatak: Analizator mrežnog protokola korištenjem Needleman Wunsch algoritma za poravnanje

Za zadanu snimljenu sjednicu analizirati mrežni protokol. Kao ulaz programa koristiti PCAP datoteku. Izlaz programa treba predstavljati lista s identificiranim poljima unutar protokola. U početku je potrebno definirati tipove polja koji se mogu nalaziti u protokolu. Isto tako potrebno je definirati zadovoljavajući rezultat (npr. postotak paketa kojima su identificirana pojedina polja). U sljedećem koraku potrebno je napraviti statistiku pojavljivanja pojedinih znakova, skupina znakova i uvjetovane vjerojatnosti pojavljivanja redoslijeda pojedinih nizova. Nakon toga potrebno je odabrati početnu listu graničnika (eng. *delimiter*) koja će se kasnije širiti novim znanjima. Na osnovu statistike definirati početnu tablicu nagrada i kazni koje će se primjenjivati pri poravnanju. Pojedine pakete je potrebno grupirati po svojstvima i statistici. Napraviti statistiku za sve pojedine tipove podataka u grupi (frekvencija pojavljivanja, duljina, graničnici). Koristeći Needleman Wunsch algoritam poravnati pakete unutar iste skupine. Na osnovu poravnanja pokušati doći do novih saznanja (npr. pojedini paketi imaju malo poravnanje s ostalima u grupi, definiranje novih graničnika i slično) i ta saznanja dodati početnima, te iterativno nastaviti postupak dok se ne dođe do zadovoljavajućeg rezultata. Isto tako definirati izlaz iz programa u slučaju da se rezultat ne poboljša nakon n koraka.

Zahvaljujem Mili Šikiću, Fakultet elektrotehnike i računarstva za stručno savjetovanje i potporu prilikom izrade ovog rada.
Također, zahvaljujem Branku Spasojeviću, studentu 3. godine Fakulteta elektrotehnike i računarstva za savjete prilikom razvoja programa.
Zahvaljujem svojoj obitelji na neizmjernoj podršci.

Sadržaj

Uvod

Prepoznavanje i analiza mrežnih protokola su vrlo važna područja za stručnjake računalne sigurnosti. Koristeći te metode moguće je otkriti sintaksu i semantiku nepoznatog protkola, ali i testirati sigurnost poznatih protokola. Za prepoznavanje elemenata protokola najčešće se koriste metode iz statistike, dok se za potrebe poravnanja nizova često koriste bioinformatički algoritmi. U ovom radu koristi se Needleman Wunsch algoritam globalnog poravnanja koji pokazuje vrlo dobra svojstva poravnanja sličnih paketa mrežnih protokola.
Osnovni cilj ovog rada je pokazati primjenjivost ovog algoritma za analizu tekstualnih mrežnih protokola, posebice HTTP protokola koji je korišten za testiranje.

U prvom poglavlju opisujem podatkovne strukture te programske alate koji se koriste u nastavku rada. Opisan je PCAP format zapisa, Wireshark alat za prisluškivanje mrežnog prometa, HTTP protokol te biblioteke korištene u programu.

Drugo poglavlje sadrži pregled svakog programskog modula zasebno. Opisani su modul za obradu ulazne datoteke, modul za grupiranje paketa te obje metode analize protkola.

Treće poglavlje predstavlja osvrt na rezultate analize protokola. Dan je ispis programa te su protumačeni rezultati izvođenja.

Četvrto poglavlje sadrži diskusiju na temu rezultata te ukazuje na moguća buduća poboljšanja programa.

U petom poglavlju je iznesen zaključak rada.

1. Podaci

Za potrebe simulacije i testiranja programa za analizu protokola korišteni su programi za prisluškivanje mrežnog prometa kako bi se sakupio testni uzorak. Kao testni protokol odabran je HTTP [10] zbog široke uporabe i jednostavne sintakse i semantike.

Prilikom izrade programa, u obzir su uzete sljedeće pretpostavke:

- Protokol koji se analizira ima format čistog teksta (eng. *plaintext*)
- Protokol nema ključnih riječi kraćih od 2 znaka
- Protokol ima ključnu riječ na prvom mjestu u paketu

1.1. PCAP

PCAP [7] je sučelje za programiranje aplikacija (eng. *Application Programming Interface*) za prisluškivanje mrežnog prometa. Unutar tog sučelja nalaze se biblioteke za pristup mrežnim uređajima te njihova sučelja prema nekoliko programskih jezika. Također sadrži i specifikaciju PCAP formata zapisa uhvaćenog mrežnog prometa. Na Unixoidnim sustavima PCAP je implementiran u libpcap biblioteci dok Microsoft Windows operacijski sustavi koriste WinPcap.

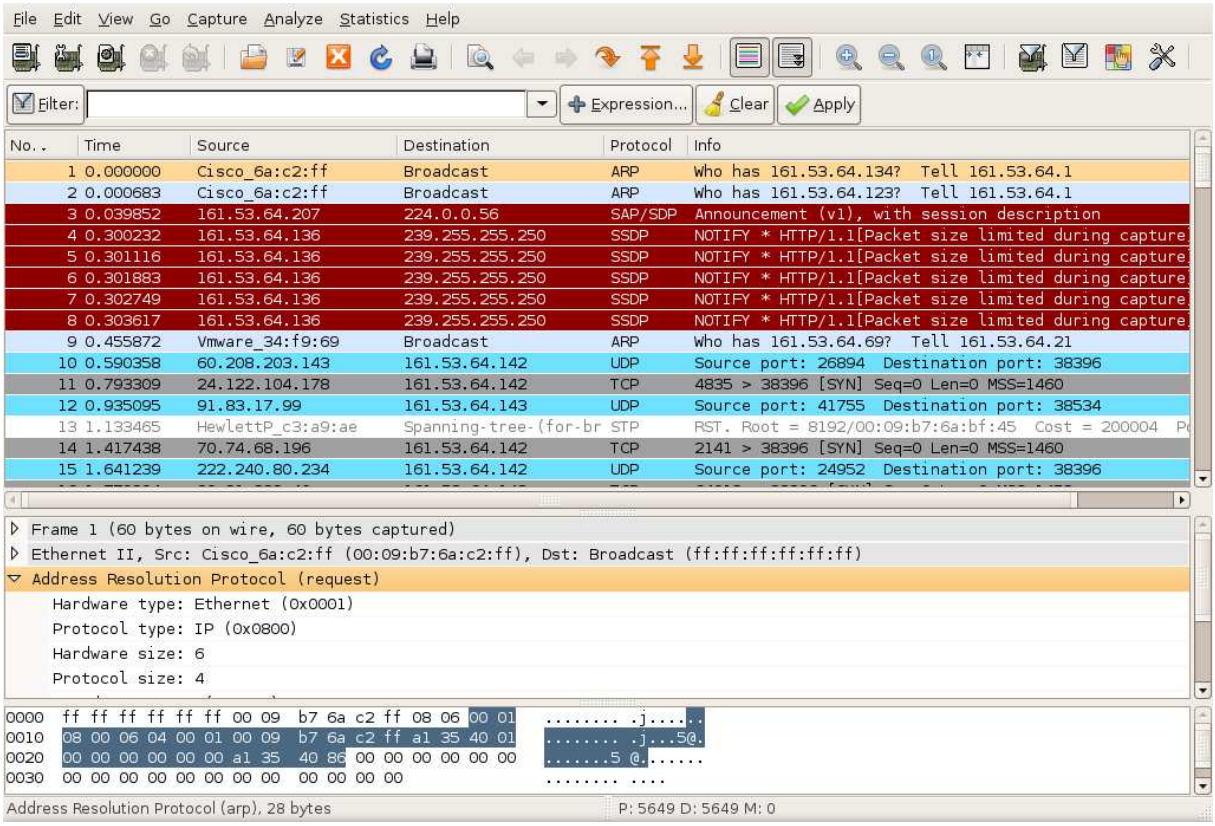
Programi za prisluškivanje mrežnog prometa koriste libpcap i WinPcap za pristup mrežnom uređaju te hvatanje paketa, a u novijim inačicama omogućeno je i slanje paketa na mrežu, spremanje uhvaćenog prometa u datoteke te čitanje prometa iz datoteke. Neki od programa koji koriste libpcap/WinPcap su tcpdump, Wireshark, Snort i nmap.

Za prikupljanje mrežnog prometa korištenog u ovom radu koristio se program Wireshark koji je opisan u nastavku.

1.1.1. Wireshark

Wireshark [8] je besplatna, višeplatformska aplikacija za prisluškivanje mrežnog prometa. Koristi se za analizu mrežnog prometa, otkrivanje grešaka u komunikaciji na mreži te razvoj i sigurnosno testiranje mrežnih protokola. Wireshark preko grafičkog korisničkog sučelja prikazuje informacije o prikupljenom prometu, a omogućuje i napredne opcije sortiranja te filtriranja prometa. Program je dostupan na Windows, Linux, Solaris te BSD operacijskim sustavima. Postavljanjem u mod za prisluškivanje (eng. *promiscious mode*) Wireshark može uhvatiti i pakete koji nisu namjenjeni računalu na kojem je pokrenut što omogućuje detaljnu analizu prometa na mreži. Koristeći uhvaćeni mrežni promet Wireshark može prepoznati i filtrirati podatke iz nekoliko stotina protokola. Dostupna je i inačica bez grafičkog korisničkog sučelja, namjenjena pokretanju iz komandnog retka, a naziva se

tshark [9]. Wireshark je licenciran pod GNU GPL (eng. *GNU General Public Licence*) licencom.



Slika 1-1. Analiza mrežnog prometa Wiresharkom

1.2. HTTP protokol

HTTP [10] (eng. *Hypertext Transfer Protocol*) je komunikacijski protokol koji omogućuje komunikaciju World Wide Web mrežom. Jedan je od najraširenijih protokola danas, a inačica HTTP/1.1 koja se i danas koristi nastala je 1999. godine. Razvoj HTTP-a nadgledaju grupe World Wide Web Consortium (W3C) i Internet Engineering Task Force (IETF), a specifikacije se izdaju u Request For Comments (RFC) dokumentima od kojih je najvažniji RFC 2616.

HTTP-om se propisuje način na koji komuniciraju klijent i poslužitelj prilikom razmjene podataka. Komunikacija započinje klijentovim zahtjevom za web izvorom nakon čega poslužitelj odgovara slanjem odgovarajućih podataka. Iako HTTP nije ovisan o TCP/IP modelu, to je ipak najčešća primjena danas u Internetu.
Najčešće ključne riječi HTTP protokola su GET, POST i HTTP.

Primjeri HTTP zahtjeva:

```
GET /index.html HTTP/1.1
HTTP/1.1 426 Upgrade Required
```

1.3. Programski alati i biblioteke

Program za analizu mrežnog protokola napisan je u programskom jeziku Python [11]. Razvoj i testiranje obavljeni su na Ubuntu Linux operacijskom sustavu s jezgrom inačice 2.6.24. Mrežni promet prikupljen je programom Wireshark s postavljenim filtriranjem prometa tako da se prikazuje samo HTTP protokol. Program tshark korišten je za pretvorbu snimljenog prometa iz PCAP formata u tekstualni oblik.

Korištene su sljedeće biblioteke:

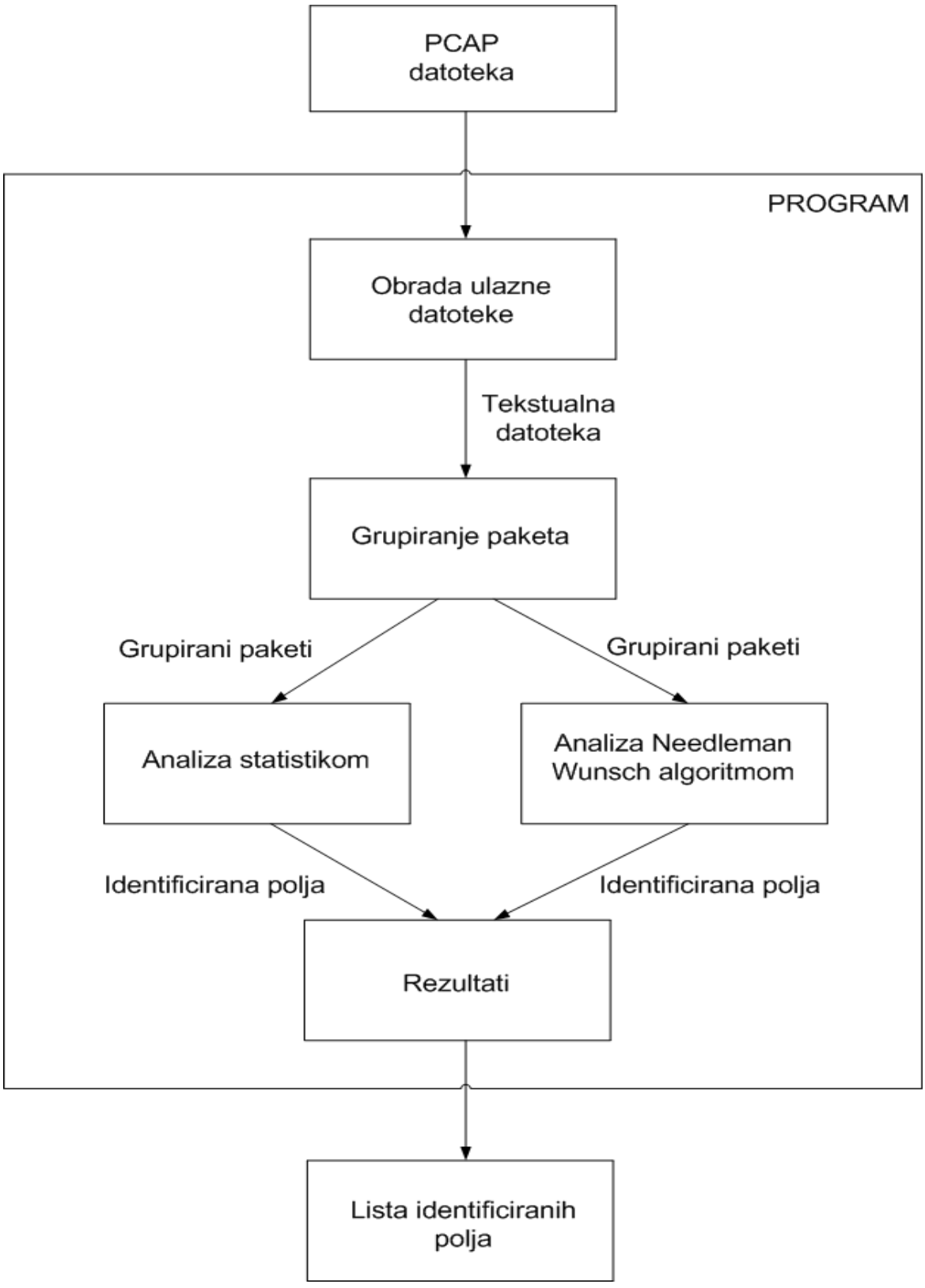
- libpcap0.8 – programska biblioteka koja omogućuje komunikaciju s mrežnim uređajima te obradu PCAP datoteka
- python-numeric – programska biblioteka za rad s matricama i naprednim matematičkim konstruktima u Python programskom jeziku

1.4. Programski zahtjevi

Sljedeći zahtjevi moraju biti ispunjeni za ispravan rad programa:

- Linux operacijski sustav s jezgrom inačice ne starije od 2.6.20.
- Instaliran Python interpreter inačice ne starije od 2.4
- Instaliran thsark program
- Instalirana biblioteka python-numeric
- Instalirana biblioteka libpcap0.8 [7]

2. Metode


PCAP
datoteka
PROGRAM
Obrada ulazne
datoteke
Tekstualna
datoteka
Grupiranje paketa
Grupirani paketi
Grupirani paketi
Analiza statistikom
Analiza Needleman
Wunsch algoritmom
Identificirana polja
Identificirana polja
Rezultati
Lista identificiranih
polja

Slika 2-1. Blok dijagram programskog rješenja

Program za analizu protokola kao ulaz uzima PCAP datoteku koja sadrži uhvaćeni mrežni promet u standardnom PCAP formatu. Program tu datoteku pretvara u tekstualnu datoteku prikladnu za obradu te zatim čita i obrađuje podatke. Kao izlaz programa daje se lista paketa s identificiranim poljima.

Program je organiziran po modulima tako da je svaki modul zadužen za jednu veću cjelinu obrade. Moduli su prikazani blok dijagramom na slici 1, a detaljnije su objašnjeni u nastavku.

2.1. Obrada ulazne datoteke

```
Obrada ulazne datoteke()
{
    Otvori PCAP datoteku
    Pretvori PCAP u CSV datoteku

    Za svaki paket:
        Izbaci nepotrebne podatke
        Upisi paket u tekstualnu datoteku

    Vrati tekstualnu datoteku
}
```

Slika 2-2. Pseudokod modula za obradu ulazne datoteke

Budući da PCAP datoteka nije prikladna za analizu paketa, potrebno ju je pretvoriti u čitljiviji oblik. Ovaj modul kao ulaz uzima PCAP datoteku, a na izlazu daje istovjetnu tekstualnu datoteku spremnu za obradu.

Koristeći program tshark, PCAP datoteka se pretvara u tekstualnu datoteku s podacima odvojenim zarezima (eng. *Comma Separated Value*). Budući da program za prikupljanje mrežnog prometa u PCAP datoteku upisuje zaglavlja koja nisu potrebna za ovu analizu, ovaj modul ta zaglavlja pronalazi i izbacuje. Nakon toga se nove vrijednosti upisuju u tekstualnu datoteku te se daljni rad programa oslanja isključivo na tu datoteku.

2.2. Grupiranje paketa

```
Grupiranje paketa()
{
    Otvori tekstualnu datoteku

    Za svaki paket:
        Rastavi paket po graničnicima
        Upisi u listu

    Pronadi kljucne rijeci
    Grupiraj pakete po kljucnim rijecima
    Vrati grupirane pakete
}
```

Slika 2-3. Pseudokod modula za grupiranje paketa

Pakete je potrebno grupirati za kasniju obradu Needleman Wunsch algoritmom i statistikom. Poravnanje paketa bitno različitih svojstava Needleman Wunsch algoritmom nema smisla jer je rezultat iznimno malo podudaranje i mala saznanja o semantici protokola. Ulaz ovog modula je lista graničnika koji se koriste za rastavljanje paketa na polja, a izlaz je lista grupiranih paketa te lista identificiranih ključnih riječi.

Prvi korak je čitanje paketa iz tekstualne datoteke te rastavljanje na polja. Polja su u paketu odvojena graničnicima te se rastavljanjem po graničnicima dolazi do svakog polja zasebno što je pogodnije za obradu.

Drugi korak je pronalaženje ključnih riječi. Algoritam pronalaženja ključnih riječi prolazi kroz sve pakete te pregledava svako polje. Za svako pojavljivanje nekog polja u tablicu zabilježi pojavu te vrijednosti. Također, sprema se i ukupan broj obrađenih paketa. Utvrđuje se minimalni postotak (granični postotak) pojavljivanja neke vrijednosti da bi se ona proglasila konstantom. Ukoliko je učestalost pojave određenog polja veća od graničnog postotka, to polje se proglašava ključnom riječi te se stavlja u listu ključnih riječi.

Nakon pronalaženja ključnih riječi slijedi grupiranje paketa po ključnim riječima. Koristi se prethodno sastavljena lista ključnih riječi. Algoritam prolazi slijedno kroz sve pakete te provjerava prvo polje u paketu koje je obavezno ključna riječ. Po prvom polju razvrstava sve pakete u grupe paketa s istom ključnom riječi na prvom mjestu. Iako svaki paket na prvom mjesu ima ključnu riječ, nisu sve ključne riječi iz liste ključnih riječi zastupljene. Točnije, ključne riječi se mogu pojaviti i na drugim mjestima u paketu, ali grupiranje se radi samo po prvoj.

Primjer paketa grupiranih po ključnoj riječi:

```
GET:
{
GET /icons/prev.jpg HTTP/1.1
GET /icons/mainmodule.jpg HTTP/1.1
}

HTTP:
{
HTTP/1.1 200 OK
HTTP/1.1 302 Moved Temporarily
}
```

2.3. Poravnanje nizova

Poravnanje nizova [5] (eng. *Sequence alignment*) koristi se u bioinformatici za poravnanje aminokiselina kako bi se otkrile sličnosti među proteinima koje mogu biti posljedica funkcionalnih, strukturnih ili evolucijskih veza među proteinima. Dopušteno je umetanje praznih znakova u bilo koji niz u svrhu boljeg poravnanja. Poravnati nizovi tipično se predstavljaju tekstualno tako da se identični znakovi nalaze u stupcima jedan ispod drugog.

2.3.1. Globalno i lokalno poravnanje

Globalno poravnanje predstavlja najbolje poravnanje preko cijele dužine dva niza. Ukoliko su nizovi različitih duljina, u kraći niz će se umetnuti praznine na odgovarajuća mjesta kako bi se duljinom izjednačio s dužim nizom. Budući da praznine narušavaju uspješnost poravnanje, njihovo umetanje se nastoji smanjiti koliko je moguće, tako je se globalnim poravnanjem uspješno poravnavaju nizovi otprilike iste duljine i sličnog sadržaja. Predstavnik globalnog poravnanja je Needleman Wunsch algoritam [1][2][6] koji je temeljen na dinamičkom programiranju (eng. *Dynamic programming*). Needleman Wunsch algoritam i dinamičko programiranje razmatraju se u nastavku rada.

Lokalno poravnanje koristi se prilikom poravnanja nizova znatno različitih duljina kod kojih očekujemo poklapanja podnizova unutar glavnog niza. Primjer algoritma za lokalno poravnanje je Smith Waterman algoritam koji se također temelji na dinamičkom programiranju. Algoritmi za lokalno poravnanje su izvan opsega ovog rada te se dalje neće razmatrati.

```
Global  FTFTALILLAVAV
        F--TAL-LLA-AV

Local   FTFTALILL-AVAV
        --FTAL-LLAAV--
```

Slika 2-4. Globalno i lokalno poravnanje

2.3.2. Dinamičko programiranje

Dinamičko programiranje [4] je metoda rješavanja preklapajućih problema (eng. *Overlapping subproblems*) koju je sredinom dvadesetog stoljeća uveo Richard Bellman. Preklapajući problemi su oni problemi koji se mogu rastaviti na manje podprobleme. Podproblemi se mogu zatim rastaviti na jos jednostavnije probleme, sve dok se ne dođe do problema koji je rješiv u realnom vremenu. Tada se optimalnim rješavanjem manjih podproblema dolazi do rješenja glavnog problema. Dinamičko programiranje može se primjeniti samo na probleme koji zadovoljavaju tzv. optimalnu strukturu (eng. *Optimal substructure*) tj. probleme za koje vrijedi da se optimalnim rješenjem manjih problema dolazi do optimalnog rješenja glavnog problema. Primjeri takvih problema su problem trgovačkog putnika i problem pronalaženja najboljeg poteza u šahu.

2.4. Analiza protokola Needleman Wunsch algoritmom

2.4.1. Teorija

Needleman Wunsch algoritam je računalno primjenjiva metoda za traženje sličnosti u nizovima aminokiselina dva proteina. Algoritam su 1970. godine objavili Saul B. Needleman i Christian Wunsch. Koristeći taj bioinformatički algoritam moguće je utvrditi postoji li sličnost između dva proteina. Maksimalna sličnost (eng. *Maximum match*) je broj koji govori o sličnosti nizova kolika je podudarnost dva niza. Jedna od njegovih definancija kaže da je to najveći broj znakova jednog niza koji se podudraju sa znakovima drugog niza a da su pritom dopušteni prekidi u oba niza. Iako mogućnost umetanja prekida rezultira velikm brojem usporedbi, Needleman Wunsch algoritam učinkovito isključuje one usporedbe koje ne pridonose maksimalnoj sličnosti. Za svaka dva niza potrebno je izračunati rezultat poravnavanja, tj. broj koji govori koliko dobro su nizovi poravnati. Shema bodovanja (eng. *scoring scheme*) je skup pravila koji pridružuje rezultat poravnanja za bilo koje poravnanje dva niza.

Prilikom poravnanja Needleman Wunsch algoritmom koriste se sljedeće podatkovne strukture:

1. Matrica zamjene (eng. *Substitution matrix*)
2. Matrica bodovanja (eng. *Score matrix*)
3. Matrica za rekonstrukciju optimalnog poravnanja (eng. *Traceback matrix*)

Matrica zamjene je matrica dimenzija M*M gdje je M broj znakova koji se mogu pojaviti u bilo koja dva niza. Ta matrica sadrži rezultat poravnanja za bilo koja dva znaka. Jednostavnije matrice koriste 1 bod ukoliko su znakovi jednaki te 0 bodova ukoliko su znakovi različiti. U bioinformatici se najčešće koriste specijalizirane matrice koje su sastavljene tako da posebno nagrade poklapanje te kazne razlikovanje znakova. Neke od poznatijih matrica zamjene su PAM250, PAM120, BLOSUM62 i BLOSUM50.

U dosadašnjem razmatranju nije uzeta u obzir razlika u duljini nizova te praznine koje pritom nastaju u kraćem nizu. Praznina odgovara umetnutom znaku, a budući da se umetanjem praznina narušava integritet niza, kazna za umetanje praznine mora biti nekoliko puta veća nego kazna za razlikovanje znakova.

Matrica bodovanja D je matrica dimenzija M*N, gdje je M broj znakova drugog niza, a N broj znakova prvog niza. Njeni elementi govore kolko je dobro podudaranje bilo kojeg znaka iz prvog niza s bilo kojim znakom iz drugog niza. Matrica se popunjava rekurzivno koristeći formulu:

$$D(i,j) = \max\begin{Bmatrix} D(i-1,j-1) + s(xi,yi) \Rightarrow diag \\ D(i-1,j) + g \Rightarrow up \\ D(i,1-1) + g \Rightarrow left \end{Bmatrix} \quad (1)$$

gdje je:
D(i,j) – element matrice bodovanja na mjestu i,j
s(i,j) – element matrice zamjene na mjestu i,j
g - kazna za prazni znak

Tabela 2-1. Matrica bodovanja

D (1,1)	D(1,2)	D(1,3)	D(1,4)	D(1,5)
D(2,1)	D(2,2)	D(2,3)	D(2,4)	D(2,5)
D(3,1)	D(3,2)	D(3,3)	D(3,4)	D(3,5)
D(4,1)	D(4,2)	D(4,3)	D(4,4)	D(4,5)

Budući da se matrica popunjava rekurzivno tj. unos u određenom polju se izračunava na temelju vrijednosti polja iznad, lijevo i dijagonalno gore-lijevo, prije ispunjavanja matrice potrebno je upisati vrijednosti u prvi redak i prvi stupac matrice. To se obavlja prilikom inicijalizacije matrice. Ispunjavanje matrice počinje poljem D(2,2) koristeći formulu 1.

Matrica za rekonstrukciju optimalnog poravnanja je matrica dimenzija M*N, gdje je M broj znakova drugog niza, a N broj znakova prvog niza. Retci matrice su znakovi prvog niza, a stupci znakovi drugog niza. Matrica se konstruira na temelju matrice bodovanja, a govori nam kako konstruirati kraći niz da bi dobili najbolje poravnanje. Rekonstrukcija počinje iz donjeg desnog polja a završava prvim lijevim poljem. To odgovara rekonstrukciji niza od zadnjeg prema prvom znaku. Prilikom rekonstrukcije, kroz matricu se možemo kretati gore („up“), lijevo („left“) ili dijagonalno gore-lijevo („diag“). Kretanje gore označava prazninu u prvom nizu, kretanje lijevo označava prazninu u drugom nizu, dok kretanje dijagonalno označava poravnanje dva znaka.

Tabela 2-2. Matrica za rekonstrukciju optimalnog poravnanja

T(1,1)	T(1,2)	T(1,3)	T(1,4)	T(1,5)
T(2,1)	T(2,2)	T(2,3)	T(2,4)	T(2,5)
T(3,1)	T(3,2)	T(3,3)	T(3,4)	T(3,5)
T(4,1)	T(4,2)	T(4,3)	T(4,4)	T(4,5)

Koraci Needleman Wunsch algoritma:

1. Određivanje matrice zamjene

2. Inicijalizacija matrice bodovanja te matrice za rekonstrukciju optimalnog poravnanja
    - Prilikom inicijalizacije matrice bodovanja, popunjavaju se prvi redak i prvi stupac. U ta polja se upisuju kazneni bodovi za praznine prilikom poravnanja. Budući da polju D(1,1) nije pridružen niti jedan znak niti jednog niza, tu se upisuje 0. Kazneni bodovi za praznine se mogu upisati kao konstantne vrijednosti za cijeli niz (npr. bilo gdje da se umetne praznina u cijelom nizu kazneni bod iznosi -3) ili kao inkrementalne vrijednosti (npr. ukoliko se praznina umeće na prvom mjestu kazneni bod iznosi -3, dok na trećem mjestu iznosi -9).

Tabela 2-3. Inicijalizacija matrice bodovanja s inkrementalnim kaznenim bodovima

0	-3	-6	-9	-12
-3	D(2,2)	D(2,3)	D(2,4)	D(2,5)
-6	D(3,2)	D(3,3)	D(3,4)	D(3,5)
-9	D(4,2)	D(4,3)	D(4,4)	D(4,5)

- Kod inicijalizacije matrice za rekonstrukciju optimalnog poravnanja također se popunjavaju samo prvi redak i prvi stupac. Rekonstrukcija niza završava kada u matrici za rekonstrukciju optimalnog poravnanja dođemo do polja T(1,1) i to polje označavamo sa „done“. Prvi redak popuni se vrijednostima „left“ budući da više nije moguće kretanje niti gore niti dijagonalno po matrici. Ukoliko prilikom procesa rekonstrukcije niza dođemo u prvi redak to znači da je rekonstrukcija drugog niza završena te će se u njega umetnuti toliko praznina koliko

je potrebno da mu se duljina izjednači s prvim. Prvi stupac matrice ispunjava se vrijednostima „up“ što označava da je prvi niz rekonstruiran te se nadopunjava prazninama.

Tabela 2-4. Inicijalizacija matrice za rekonstrukciju optimalnog poravnanja

done	left	left	left	left
up				
up				
up				

3. Izračun vrijednosti matrice bodovanja te popunjavanje matrice za rekonstrukciju optimalnog poravnanja
    - Matrica bodovanja popunjava se slijedno počevši od polja D(2,2), zatim D(2,3), D(2,4) itd. Nakon što se izračuna vrijednost matrice bodovanja za određena 2 znaka, u matricu rekonstrukcije se u isto polje upisuje vrijednost uz izraz koji je imao maksimalnu vrijednost.

4. Proces rekonstrukcije poravnatog niza
    - Koristeći matricu rekonstrukcije rekonstruiraju se oba optimalno poravnata niza. Rekonstrukcija kreće iz donjeg desnog polja, a završava u gornjem lijevom polju označenom „done“. Pritom „diag“ označava da su znakovi poravnati te se oni prepisuju jedan ispod

drugog. „up“ označava prazninu u gornjem nizu (niz čiji znakovi predstavljaju stupce matrice) te se u taj niz upisuje praznina dok se znak iz drugog niza prepisuje. „down“ označava prazninu u lijevom nizu (niz čiji znakovi predstavljaju retke matrice) te se u taj niz upisuje praznina dok se znak iz drugog niza prepisuje.

Tabela 2-5. Primjer rekonstrukcije nizova

done	left	left	left	left
up	diag	left	left	left
up	diag	diag	diag	left
up	up	up	diag	diag

2.4.2. Realizacija

```
Analiza Needleman Wunsch algoritmom()
{
    Za svaki paket:
        Poravnaj pakete Needleman
        Wunsch algoritmom

    Identificiraj zajednicke podnizove
    Ukloni duplikate
    Vrati identificirane pakete
}
```

Slika 2-5. Pseudokod modula za analizu koristeći Needleman Wunsch algoritam

Dio programa koji obavlja analizu koristeći Needleman Wunsch algoritam programski je realiziran unutar modula nw.py. Modul kao ulaz uzima listu paketa grupiranih po ključnim riječima, a kao izlaz vraća listu paketa s identificiranim poljima.

Prvi korak je poravnanje nizova Needleman Wunsch algoritmom radi otkrivanja mjesta podudaranja. Budući da je Needleman Wunsch algoritam procesorski zahtjevan poravnanje se ne obavlja između svaka 2 paketa unutar iste grupe. To također nije efikasno jer poravnanjem dva paketa iz različitih sjednica nebi dobili nikakva dodatna saznanja o semantici protokola zbog prevelike različitosti. Također, duljine paketa između sjednica mogu jako varirati što je nepovoljno jer se usporedbom 2 paketa znantno različitih duljina dobije jako malo informacija. Poravnanje se obavlja tako da se svaki paket poravnava sa sljedećim osim zadnjeg paketa koji se poravnava s prvim. Prilikom programske realizacije Needleman Wunsch algoritma za spremanje podataka u matrice korišten je Python Numeric programski modul za lakše zapisivanje i rukovanje matricama unutar programa. Korištena je jednostavna matrica zamjene, gdje se podudaranje boduje s jednim bodom, razlikovanje s 0 bodova, a kazneni bodovi za praznine su inkrementalni i stupanj povećanja iznosi -6 bodova. Rezultat ovog koraka su poravnati paketi.

Sljedeći korak je prepoznavanje istih sekvenci unutar poravnatih nizova. Cilj ovoga je otkrivanje mogućih podudarnosti protokola na određenim mjestima. Ukoliko više paketa na istom mjestu ima isti znakovni podniz, može se zaključiti da to nije slučajna vrijednost te se to saznanje iskoristi prilikom sastavljanja ispravnog paketa protokola tako da se ta vrijednost koja se pojavljuje u više paketa na istom mjestu prepiše, a vrijednosti koje variraju od paketa do paketa se unose proizvoljno. Usporedba se obavlja nad nizovima poravnatim Needleman Wunsch algoritmom, tako da se međusobno uspoređuju upravo oni paketi koje je Needleman Wunsch algoritam poravnao. Usporedba se vrši znak po znak kroz cijelu dužinu poravnatih paketa. Budući da su poravnati Needleman Wunsch algoritmom, paketi će biti jednakih duljina. Prilikom usporedbe, gleda se da li

nizovi na istom mjestu imaju isti podniz. Ukoliko se nizovi na istom mjestu poklope u jednom znaku, program će to interpretirati kao konstantnu vrijednost, što nije dobro jer se tako stvara previše lažnih konstanti. Zato je uvedena minimalna duljina podniza te se paketi moraju poklopiti u najmanje toliko znakova za redom da bi se podniz interpretirao kao konstanta. Eksperimentiranjem je zaključeno da je najbolja vrijednost za minimalnu duljinu podniza 2 znaka. Ukoliko paketi imaju na istom mjestu isti podniz duljine najmanje 2 znaka, on će se interpretirati kao konstanta. Inače je varijabla. Rezultat ovog koraka su identificirana polja unutar paketa.

Treći korak je određivanje tipa identificiranih polja u paketu. Ovo je potrebno da bi se kasnije prilikom rekonstrukcije paketa moglo što točnije stvoriti ispravan paket. Tipovi polja koji su podržani su:

- Znakovni niz (String) – u paketu se označava sa STR
- Cijeli broj (Integer) – u paketu se označava s INT
- Broj s pomičnim zarezom (Float) – u paketu se označava s FLOAT

Uz svako identificirano polje upisuje se oznaka da li je polje konstana („CONST“) ili varijabla („VAR“). Ukoliko je konstanta upisuje se i vrijednost konstante, a ukoliko je varijabla, upisuje se tip polja („STR“, „INT“, „FLOAT“).

Posljednji korak je uklanjanje duplikata iz liste paketa. Prilikom prisluškivanja prometa na mreži moguće je da se uhvati više puta isti paket, što u konačnici rezultira s više istih paketa u listi identificiranih paketa. Budući da nije potrebno znati koliko je takvih paketa bilo, oni se izbacuju iz liste identificiranih.

Primjer ispisa modula za identifikaciju polja koristeći Needleman Wunsch algoritam:

```
„CONST:POST /mail/?ui=2“, „VAR:INT“, „CONST:HTTP/1.1“
„CONST:GET /mail/“, „VAR:STR“, „CONST:4524 HTTP/1.1“
```

2.5. Analiza protokola statistikom

```
Analiza statistikom()
{
    Za svaki paket:
        Pregledaj cijeli paket za
        kljucne rijeci
        Zamijeni polja u paketu kljucnim
        rijecima ili varijablama

    Umetni graničnike
    Ukloni duplikate
    Vrati identificirane pakete
}
```

Slika 2-6. Pseudokod modula za analizu statistikom

Dio programa koji obavlja analizu koristeći statistiku programski je realiziran unutar modula statistics.py. Modul kao ulaz uzima listu paketa grupiranih po ključnim riječima, a kao izlaz vraća listu paketa s identificiranim poljima.

Budući da su ključne riječi protokola poznate, potrebno je proći kroz sve pakete u ulaznoj listi neidentificiranih paket i za svako polje odrediti radi li se o konstanti ili varijabli. Ukoliko se radi o varijabli, potrebno je dodatno odrediti i kojeg je tipa podatka ta varijabla. Detekcija konstanti radi se tako da se provjerava da li vrijednost polja u listi ključnih riječi koja je primljena kao ulazni argument module. Ukoliko je radi se o ključnoj riječi protokola odnosno konstanti te se na to mjesto upisuje „CONST“ i vrijednost ključne riječi, npr. „CONST:GET“. Ukoliko neko polje u paketu u listi ključnih riječi znači da se radi o varijabli. Za varijablu se upisuje riječ „VAR“ te se određuje tip podatka koji se može nalaziti u varijabli. Mogući tipovi podataka su „INT“, „FLOAT“ i „STR“. Tip podatka se određuje na temelju sadržaja tog polja i to na način da se vrši pretvorba tipova podataka. Prvo se podatak pokuša pretvoriti u cjeli broj i ukoliko se pretvorba obavi bez gubitka informacije, radi se o cijelom broj. Na isti način određuju se tipovi podataka za brojeve s pomičnim zarezom i znakovne nizove. Nakon što se odredi tip podatka upisuje se iza riječi „VAR“, npr. „VAR:INT“.

Nakon detekcije tipova polja unutar svakog paketa između polja se umeću graničnici po kojima su polja prvotno bila rastavljena, radi kasnijeg lakšeg sastavljanja paketa.
Zadnji korak je uklanjanje duplikata iz liste identificiranih polja.

Primjer ispisa modula za analizu koristeći statistiku:
„CONST:POST“, „VAR:INT“, „CONST:HTTP/1.1“
„CONST:GET“, „VAR:STR“, „CONST:HTTP/1.1“

3. Rezultati

Rezultati prepoznavanja biti će posebno razmotreni za obje metode analize protokola. Testna PCAP datoteka sadrži snimljenu sjednicu od 132 HTTP paketa. Redoslijed paketa nije mjenjan, paketi se obrađuju u onom redoslijedu u kojem su uhvaćeni.

Testno izvođenje programa obavljeno je na osobnom računalu s procesorom snage 1.73GHz te jednim gigabajtom radne memorije.

3.1. Rezultati analize statistikom

Izvođenje analize statistikom koristeći testnu PCAP datoteku na testnom računalu trajalo je 1.2 sekunde. Postotak potreban da se polje identificira kao ključna riječ postavljen je na 10%.

Program je prepoznao i identificirao sljedeće HTTP pakete:
['CONST:POST', ' ', 'VAR:STR', ' ', 'CONST:HTTP/1.1']
['CONST:GET', ' ', 'VAR:STR', ' ', 'CONST:HTTP/1.1']
['CONST:HTTP/1.1', ' ', 'CONST:200', ' ', 'CONST:OK']
['CONST:HTTP/1.1', ' ', 'VAR:INT', ' ', 'VAR:STR', ' ', 'VAR:STR']

Iz rezultata je vidljivo da je program prepoznao najvažnije semantičke konstrukte HTTP protokola te identificirao tipove polja za varijable. Iz gore navedenih nizova se uspješno mogu rekonstruirati ispravni HTTP paketi tako da se konstante prepišu a na mjesto varijabli umetne proizvoljna vrijednost ovisno o tipu podatka.

Rezultati se interpretiraju na sljedeći način:
Prvi rezultat otkriva semantiku ispravnog HTTP POST zahtjeva. Na prvom mjestu nalazi se ključna riječ POST, zatim dolazi proizvoljni znakovni niz koji označava resurs koji se traži od poslužitelja te na kraju se nalazi inačica HTTP protokola kojom se obavlja komunikacija. Umetnute praznine iza ključne riječi POST te iza varijable označavaju da se tu nalazi graničnik protokola, u ovom slučaju praznina. Koristeći rezultat programa konstruiramo sljedeći potpuno ispravan HTTP zahtjev:

```
POST proizvoljan_niz HTTP/1.1
```

Drugi rezultat prikazuje semantiku HTTP GET zahtjeva koja je vrlo slična POST zahtjevu pa neće biti detaljnije objašnjena.

Treći rezultat otkriva semantiku HTTP statusne poruke. Na prvom mjestu se nalazi protokol kojim se obavlja komunikacija, na drugom mjestu je standardni HTTP kod koji označava uspješno obavljenu komunikaciju te na kraju slijedi ključna riječ „OK“

što je slovčana oznaka za kod 200. U ovom slučaju su sva polja konstante te se jednoznačno može rekonstruirati HTTP paket:

```
HTTP/1.1 200 OK
```

Posljednji rezultat također otkriva semantiku HTTP statuse poruke. U ovom slučaju nije bilo dovoljno pojavljivanja ove statusne poruke te program nije točno mogao odrediti o kojim se ključnim riječima radi, već su protumačene kao varijable. Primjeri ovih paketa su:

```
301 Moved Permanently
204 No Content
```

Smanjenje postotka potrebnog da se određeno polje identificira kao ključna riječ rezultira prepoznavanjem statusnih poruka koje nisu uspješno prepoznate u četvrtom rezultatu, ali se javlja puno više lažno prepoznatih polja pa je optimalna vrijednost za taj postotak 0.1 odnosno 10%.

3.2. Rezultati analize Needleman Wunsch algoritmom

Izvođenje analize Needleman Wunsch algoritmom koristeći testnu PCAP datoteku na testnom računalu trajalo je 4.5 sekundi. Minimalan broj znakova potreban da se niz identificira kao konstanta je postavljen na 2.

Program je prepoznao i identificirao sljedeće pakete:

```
['CONST:POST   /mail/?ui=2&ik=7d3935389e&view',   'VAR:STR',
'CONST:=au&rt=j', 'VAR:STR ', 'CONST: HTTP/1.1']
['CONST:GET /', 'VAR:STR ', 'CONST:cco/', 'VAR:STR ', 'CONST:
HTTP/1.1']
['CONST:GET /mail/', 'VAR:STR ', 'CONST: HTTP/1.1']
['CONST:GET /mail/', 'VAR:STR ', 'CONST:ieage', 'VAR:STR ',
'CONST:ser&v', 'VAR:STR ', 'CONST:icy HTTP/1.1']
['CONST:GET /mail/', 'VAR:STR ', 'CONST:image', 'VAR:STR ',
'CONST:vico', 'VAR:STR ', 'CONST:.ico HTTP/1.1']
['CONST:HTTP/1.1 304 Not Modified']
['CONST:HTTP/1.1 200 OK']
['CONST:HTTP/1.1 302 Moved Temporarily']
```

Rezultat programa je 70 nizova sličnih gore navedenima, ali je zbog preglednosti navedeno samo nekoliko primjera.
Ispis ovog dijela analize znatno se razlikuje od analize statistikom. Kao što je vidljivo, analiza Needleman Wunsch algoritmom prepoznala je i sadržaj unutar varijabli.

Rezultati se interpretiraju na sljedeći način:
Kao i kod rezultata analize statistikom, na prvom mjestu nalaze se ključne riječi, no u ovom slučaju polja nisu odjeljena graničnikom protokola, već se jedno polje može sadržavati proizvoljan broj graničnika. Prvi rezultat nam govori kako se kod više paketa javlja ključna riječ POST iza koje slijedi niz znakova te je program to protumačio kao konstantnu vrijednost. U ovom slučaju taj niz znakova označava identifikaciju sjednice komunikacije između klijenta (web preglednika) i poslužitelja.

Nakon toga slijedi varijabla te ponovo konstantni niz znakova. Iz gore navedenih rezulata konstruiramo potpuno ispravne HTTP zahtjeve:

```
POST  /mail/?ui=2&ik=7d3935389e&viewneki_znakovni_niz=au&rt=j
HTTP/1.1
GET /mail/neki_znakovni_niz HTTP/1.1
```

Iz zadnja 3 rezultata vidljivo je da je analiza Needleman Wunsch algoritmom prepoznala i HTTP statusne poruke koje analiza statistikom nije uspjela. Točnije, Needleman Wunsch zahtjeva manje pojavljivanja određenih podnizova da se identificiraju kao konstante.

Iz rezultata je vidljivo da analiza Needleman Wunsch algoritmom daje više informacija o semantici protokola jer analizira i sadržaj varijabli, a ne samo mjesta na kojima se pojavljuju varijable. Koristeći rezultate Needleman Wunsch algoritma, moguće je sastaviti zahtjeve vrlo slične onima iz ulazne datoteke što omogućuje efikasniju provjeru sigurnosti protokola.

4. Diskusija

Rezultati su pokazali kako je moguće uspješno programski analizirati protokol te identificirati polja uz vrlo mali uzorak snimljenog mrežnog prometa. Rezultati analize Needleman Wunsch metodom daju detaljan i opširan prikaz semantike protokola uz vrlo malo promašaja, tj. krivo prepoznatih nizova. U ovako malom uzorku uspješno je prepoznato više od 50% paketa. Rezultati analize statistikom ne odaju toliko informacija, ali su korisni za saznavanje sintaksne strukture protokola te kontrolu rezultata dobivenih Needleman Wunsch analizom. Na većem uzorku snimljenih paketa može se očekivati blago poboljšanje rezultata analize statistikom, ponajprije u broju prepoznatih paketa.

Program je moguće poboljšati implementiranjem automatske metode otkrivanja graničnika u protokolu te naprednijih metoda u dijelu programa koji se bavi grupiranjem paketa. Korisno bi bilo implementirati podršku za analizu binarnih protokola, koja bi omogućila analiziranje šireg sprektra protokola, uključujući i sve raširenije zatvorene vlasničke (eng. *proprietary*) protokole. Implementiranjem automatske metode utvrđivanja graničnika u protokolu dobila bi se neovisnost programa o samom protokolu te mogućnost analize nepoznatih i protokola s više različitih graničnika. Nepoznavanje svih graničnika protokola može rezultirati pogrešnim grupiranjem paketa po ključnim riječima te potpuno promašenim rezultatima obje analize pa je ispravno prepoznavanje graničnika u protokolu neophodno za ispravan rad programa. Implementacijom naprednijih metoda u dijelu za grupiranje paketa poboljšala bi se prvenstveno metoda analize Needleman Wunsch algoritmom jer bi na osnovu grupiranih paketa bilo moguće bolje poravnati pakete unutar iste grupe te tako dobiti bolje prepoznavanje.

5. Zaključak

Cilj ovog rada iz područja računalne sigurnosti je bila realizacija programa za analizu tekstualnih mrežnih protokola. Korištene su dvije različite metode: analiza statistikom te analiza Needleman Wunsch algoritmom. U radu je dana kratka teoretka podloga korištenih metoda, dok je praktični dio napravljen korištenjem Python skripti. Za provjeru i testiranje programa korišteni su stvarni podaci prikupljeni specijaliziranim programima.

U ovom radu sam pokazao da je Needleman Wunsch algoritam uspješno primjenjiv alat za analizu protokola u računalnoj sigurnosti. Pokazalo se da je dovoljan vrlo mali uzorak mrežnog prometa za točnu rekonstrukciju većine paketa određenog tekstualnog protokola. Rezultati analiziranja HTTP protokola su pokazali da je prepoznavaje u potpunosti uspjelo. Iako su rezultati prepoznavanja testnog protokola vrlo dobri, za prepoznavanje složenijih protokola potrebno je implementirati dodatne funkcionalnosti. Needleman Wunsch algoritam se pokazao kao vrlo kvalitetan alat za analizu te uz daljnji rad na programu očekujem vrlo dobre rezultate analize drugih protokola te uspješnu primjenu programa u domeni računalne sigurnosti.

6. Literatura

[1] Likić, V: The Needleman-Wunsch algorithm for sequence alignment, 7th Melbourne Bioinformatics Course

[2] Wikipedia stranica Needleman-Wunsch algoritma, http://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm, 20.5.2008.

[3] Basic-Algorithms-of-Bioinformatics Applet, http://baba.sourceforge.net/, 2.6.2008.

[4] Wikipedia stranica dinamičkog programiranja, http://en.wikipedia.org/wiki/Dynamic_programming, 6.6.2008.

[5] Wikipedia stranica algoritama za poravnanje nizova, http://en.wikipedia.org/wiki/Sequence_alignment, 1.6.2008.

[6] Needleman, S.B; Wunsch C.D: A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins, srpanj 1969.

[7] Wikipedia stranica libpcap biblioteke, http://en.wikipedia.org/wiki/Libpcap, 26.5.2008.

[8] Web stranica Wireshark alata, http://www.wireshark.org/, 15.5.2008.

[9] Web stranica tshark alata, http://www.wireshark.org/docs/man-pages/tshark.html, 16.5.2008.

[10] Web stranica HTTP protokola, http://www.w3.org/Protocols/, 18.5.2008.

[11] Web stranica Python Numeric biblioteke, http://numpy.scipy.org/, 11.5.2008.

Analizator mrežnog protokola korištenjem Needleman Wunsch algoritma za poravnanje

Cilj ovog rada iz područja računalne sigurnosti je bila realizacija programa za analizu tekstualnih mrežnih protokola. Korištene su dvije različite metode: analiza statistikom te analiza Needleman Wunsch algoritmom. U radu je dana kratka teoretka podloga korištenih metoda, dok je praktični dio napravljen korištenjem Python skripti. Za provjeru i testiranje programa korišteni su stvarni podaci prikupljeni specijaliziranim programima.

Ključne riječi: analiza, protokol, statistika, Needleman Wunsch, bioinformatika, računalna sigurnost

Analysis of network protocol using Needleman Wunsch sequence alignment algorithm

The goal of this computer security research was implementation of network protocol analysis program. Methods used to analyse protocol are statistics and Needleman Wunsch algorithm. Brief theoretical background is laid in this paper while the program itself is coded in Python programming language. Testing and verifying program output was conducted using real network data gathered with specialized programs.

Keywords: analysis, protocol, statistics, Needleman Wunsch, bioinformatics, computer security