Sieťové aplikácie a správa sieti

Prenos súboru skrz skrytý kanál

8. novembra 2021

Peter Zdravecký (xzdrav00)

Obsah

Úvod

Zadaním projektu je implementovať prenos súborov skrz skrytý kanál. Tento proces posielania súborov sa uskutočňuje pomocou protokolu `ICMP` a jeho `Echo-Request` správ. Program sa skladá z dvoch častí:

- **klientska časť**, ktorá slúži na odosielanie súborov v šifrovanej podobe,
- **serverová časť**, ktorá slúži na prijímanie odosielaných súborov, dešifrovanie a ukladanie na disk.

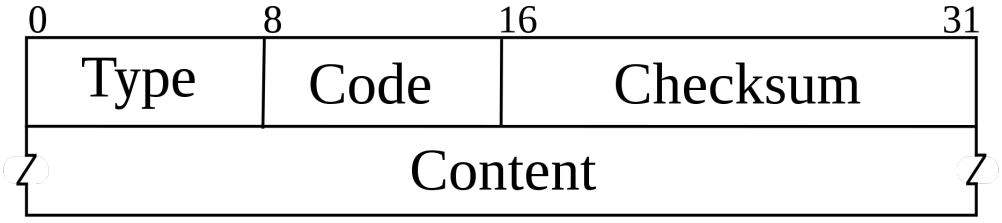
Program podporuje komunikáciu na sieťovej vrstve pomocou protokolov `IPv4` aj `IPv6`. Prenos medzi klientom a serverom je považovaný za spoľahlivý a pri strate paketu alebo neúplnom odoslaní, sa súbor považuje za poškodený.
V rámci vypracovania projektu bolo implementovaných niekoľko rozšírení nad rámec základného zadania.

- server je schopný prijímať viac súborov súčasne od viacerých klientov,
- klient je schopný prijať na vstup programu názov zložky, z ktorej sa rekurzívne odošlú všetky súbory na danú adresu.
- implementácia `Wireshark dissectoru`[1] pre vlastný protokol viz sekcia 1.3

Teória

Ako to bude fungovať?

V sieti sa protokol `ICMP`[1] využíva na odosielanie chybových správ alebo oznámení. Príkladom takej správy môže byť, že daná služba nieje dostupná alebo router nieje dosiahnuteľný. Takto vyzerá štruktúra `ICMP` paketu typu `Echo-Request` (ping):



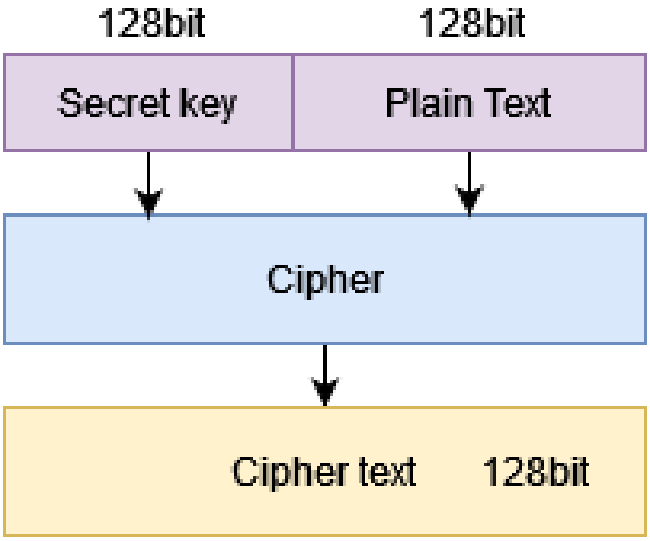
Obr. 1: Štruktúra ICMP paketu

Ako si možno všimnúť, hlavičku `ICMP` paketu tvorí prvých 64 bitov (8B) a zvyšok je voľný pre akékoľvek dáta. Táto skutočnosť bude využitá pre prenos súboru. Zašifrované dáta budeme odosielať v tejto časti paketu. Jediné obmedzenie pri odosielaní paketu je jeho veľkosť. Maximálna veľkosť prenášaného paketu je definovaná pomocou **MTU** (Maximum transmission unit) - t.j. maximálna prenosová jednotka po sieti, ktorá je bežné nastavená na 1500B. Preto budú väčšie súbory rozdelené do viacerých paketov.

[1] https://www.wireshark.org/docs/wsdg_html_chunked/ChapterDissection.html

Šifrovanie

Pre šifrovanie dát zo súboru sa využíva šifra **AES** (Advanced Encryption Standard)[4]. Je to technika *symetrického šifrovania*, čo znamená, že jeden kľúč je využívaný na šifrovanie, ale aj na dešifrovanie dát. Táto šifra využíva šifrovanie po blokoch o veľkosti 128 bitov (16B).



Obr. 2: Názorná ukážka šifrovania

1 Návrh

Program je napísaný v jazyku `C++` a využíva sa v ňom objektovo orientované programovanie. Využíva základné knižnice štandardu a knižnicu `pcap`[2]. Celý program je implementovaný v 5 moduloch:

- `main`,
- `icmp_client`,
- `icmp_server`,
- `cipher`,
- `secret_proto`.

V module `main` prebieha spracovávanie argumentov príkazového riadka a vytvorenie inštancií tried `icmp_client` a `icmp_server`. Podľa vstupu z príkazového riadku sa následne vyberie inštancia, s ktorou sa bude pracovať.

1.1 Klientská časť

Slúži na odosielanie súboru/súborov v šifrovanej podobe po sieti. Táto časť programu je vypracovaná v module `icmp_client`, ktorý obsahuje implementáciu rovnomennej triedy.

Trieda `icmp_client` obsahuje konfiguráciu klientskej časti, napr. ID klienta, ktorý slúži na identifikáciu spojenia so serverom, názov odosielaného súboru, informácie o destinácii, kam sa má súbor odoslať, nastavenie socketu...

Postupnosť akcií, ktoré program vykonáva v klientskej časti:

1. na základe argumentu, ktorý popisuje destináciu (názov destinácie alebo jej IP adresa), zistí požadované informácie o destinácii kam sa súbor bude odosielať a pomocou akého sieťového protokolu (`IPv4` / `IPv6`),
2. nakonfiguruje socket a otvorí súbor ktorý sa bude odosielať,
3. odošle prvý paket ktorý oboznámi server o zahájení prenosu odosielania súboru,
4. začne odosielať pakety so šifrovanými dátami,
5. po odoslaní všetkých dát odošle posledný paket na server ktorý oboznámi o ukončení prenosu dát.

1.2 Serverová časť

Slúži na zachytávanie odosielaných súborov, dešifrovanie a ukladanie na disk. Sever je schopný prijímať viac súborov súčasne pomocou mapovania jednotlivých prenosov na základe identifikátora klienta. Táto časť programu je vypracovaná v module `icmp_server`, ktorý obsahuje implementáciu rovnomennej triedy.

Trieda `icmp_server` obsahuje:

- informácie o rozhraní, na ktorom sa zachytávajú odosielané dáta,
- dátovú štruktúru typu `map`, ktorá slúži na mapovanie jednotlivých spojení.

Tieto spojenia sú definované pomocnou štruktúrou, ktorá ma názov `fileinfo` a obsahuje:

- ukazovateľ na súbor, do ktorého sa zapisuje,
- sekvenčné číslo, ktoré slúži na kontrolu spoľahlivého prenosu,
- vektor, do ktorého sa ukladajú dáta na zapisovanie.

Postupnosť akcií, ktoré sa vykonávajú v serverovej časti:

1. konfigurácia serveru pomocou volania funkcie `init()`, ktorá zabezpečí správne nastavenie rozhrania na ktorom sa búde počúvať a nastaví filter pre odchytávanie `ICMP` paketov,
2. server začne zachytávať pakety,

3. čaká na prijatie paketu ktorý oznamuje o zahájení prenosu a ktorý obsahuje názov prenášaného súboru,
4. pripraví súbor na zapisovanie,
5. zachytáva odosielané dáta, dešifruje ich a zapisuje do súboru,
6. po prijatí paketu oznamujúceho o ukončení odosielania dát, zatvorí súbor a vypíše hlásenie o úspešnom prijatí súboru.

[Dôležité] Ukončenie serveru sa realizuje pomocou klávesovej skratky `Ctrl + C`

1.3 Šifrovanie a vymyslený protokol

Modul `cipher` obsahuje funkcie na šifrovanie a dešifrovanie dát. Hlavičkový súbor `secret_proto.h` obsahuje definíciu protokolu, ktorý uchováva podstatné informácie.

```
enum pkt_type { HEAD, DATA, END };

struct secret_proto {
    char proto_name[4] = "MNT";
    int type;
    int datalen;
    int seq;
    int client_id;
};
```

Tento vymyslený protokol slúži na odlíšenie bežných `ICMP` paketov od paketov, ktoré sú odosielané z klientskej časti. Na filtrovanie slúži názov protokolu `MNT`. Popis ďalších premenných štruktúry `secret_proto`:

- `type` slúži na rozoznanie paketu, či sa jedná o paket ktorý začína spojenie, paket ktorý obsahuje dáta, alebo paket ktorý ukončuje spojenie,
- `datalen` obsahuje dĺžku prenášaných dát,
- `seq` obsahuje sekvenčné číslo paketu, ktorý je odosielaný; sekvenčné číslo slúži na zabezpečenie spoľahlivého prenosu,
- `client_id` slúži na identifikáciu spojenia pomocou identifikátoru klienta

Pre tento protokol bol vytvorený `dissector` ktorý slúži na odlíšenie bežných `icmp` paketov od paketov ktoré sú odosielané z klientskej časti. Implementácia sa nachádza v súbore `secret_proto.lua`.

2 Implementácia

2.1 Spracovanie argumentov

Pre spracovanie argumentov je využitá knižnica `getopt`[3]. Dostupné prepínače pri spúštaní programu:

```
[] = required argument, {} - optional argument
[-r file|folder]
[-s ip|hostname]
{-l}
{-h}
```

Prepínač `-r` slúži na určenie súboru/zložky ktorý sa bude prenášať. Prepínač `-s` slúži na definovanie IP adresy alebo názvu destinácie kam sa súbor pošle. Prepínač `-l` slúži pre zapnutie programu v režime server. Pri použití prepínaču `-l` nieje nutné zadávať prepínač `-s` a `-r`.

2.2 Popis funkcií modulu `icmp_client`

`get_dest_info()`

Získava 2 základné informácie o destinácii na základe argumentu, ktorý bol zadaný v príkazovom riadku: verzia sieťového protokolu, ktorým sa bude posielať, a IP adresu destinácie. Na získanie týchto informácii bola použitá funkcia `getaddrinfo()`. Následne sa zo získaných informácia vytvára socket, pomocou ktorého sa bude komunikovať.

`prepare_file()`

Overuje existenciu súboru. Ak súboru existuje, funkcia sa ho pokúsi otvoriť.

`get_file_data()`

Číta dáta zo súboru. Na vstup do funkcie prichádza dĺžka dát, ktorá sa má zo súboru prečítať. Funkcia vracia ukazovateľ na dáta, ktoré boli prečítané zo súboru. Ďalším vstupom funkcie je ešte ukazovateľ na celé číslo, do ktorého sa zapisuje skutočný počet prečítaných dát. V prípade, že v súbore je menej dát ako sa snažíme precitať, tak prečítame všetko čo ostalo.

`create_packet()`

Zostavuje `ICMP` paket a dátovú časť. V tejto funckii sa alokuje pamäť potrebná pre vytvorenie paketu. Vytvorí sa hlavička `ICMP` paketu a za ňou sa vytvorí hlavička pre náš vymyslený protokol `secret_proto`. Zvyšok paketu sa zaplní dátami. Funkcia vracia ukazovateľ na vytvorení paket.

`csum()`

Slúži na vypočítanie kontrolnej sumy pre `ICMP` hlavičku.

`send_pkt()`

Slúži na posielanie paketov. V tejto funkcii sa vyplnia informácie o odosielanom pakete, šifrujú sa dáta na odoslanie, volá sa funkcie na zostavenie paketu a následne sa tento paket posiela po sieti.

`send_file()`.

Hlavná funkcia programu, ktorý volá pomocné funkcie na inicializáciu a správne fungovanie programu. Na vstup do funkcie ide názov súboru a destinácia, kam sa ma súboru odoslať. Funkcia odosiela prvotný paket na zahájenie komunikácie pomocou funkcie `send_pkt()`, následne sa prečítajú, šifrujú dáta zo súboru a posielajú po sieti. Po prenesení všetkých dát zo súboru sa posiela posledný paket na ukončenie spojenia. Funkcia uvoľní všetky alokované zdroje a vypíše hlásenie o úspešnom poslaní súboru na určenú destináciu.

2.3 Popis funkcií modulu `icmp_server`

`init()`

Slúži na konfiguráciu základných hodnôt potrebných k spusteniu zachytávania paketov v sieti. Pomocou funkcie `pcap_findalldevs()` sa určí prvé dostupné ethernetové rozhranie, na ktorom sa bude počúvať. Po vykonaní tejto funkcie sa spustí funkcia `pcap_open_live()`, ktorá získa popisovač zachytávania paketov. Filter pre zachytávanie `ICMP` paketov v textovej podobe sa preloží do filtrovacieho programu pomocou funkcie `pcap_compile()` a nastaví pomocou `pcap_setfilter()`.

`start()`

Server prejde pomocou tejto funkcie do stavu, kedy zachytáva pakety, ktoré sa následne spracúvajú vo funkcii `handle_packet()`.

`handle_packet()`

Spracuje prijatý paket. Prečíta ethernetovú hlavičku paketu a rozhodne, o akú verziu sieťového protokolu sa jedná. Následne na základe verzie protokolu volá funkciu, ktorá spracováva ďalšiu vrstvu paketu.

`handle_data()`

Táto funkcie je určená na spracovanie L4 vrstvy paketu, kde sú uložené dáta, ktoré klient odosiela. Najprv je nutné skontrolovať, či je prijatý paket odoslaný z klientskej časti. To je možné pomocou daného vymysleného protokolu, kde je uložené jeho meno. Iné pakety sa zahadzujú. Táto funkcia taktiež overuje, či sa jedná o `echo-request` typ `ICMP` protokolu, ostatné pakety zahadzuje. Po overení, že paket bol odoslaný z klientskej aplikácie sa prijaté dáta dešifrujú a na základe typu paketu, ktorý sme zistili z vymysleného protokolu sa volá ďalšia funkcia:

- funkcia `new_file()` na vytvorenie nového spojenia,
- funkcia `file_write()` na zapísanie dát do súboru,
- funkcia `file_transferd()` na ukončenie spojenia.

`new_file()`

Vytvára nové spojenie, ktoré sa mapuje na identifikačné číslo klienta. Otvára súbor, do ktorého sa budú zapisovať prijaté dáta.

`file_write()`

Slúži na zapisovanie prijatých dát do súboru. Na základe identifikačného čísla klienta sa vyberie súbor, do ktorého sa máju zapísať dáta. Z dôvodu obmedzenia I/O volaní a zvýšenia rýchlosti programu sa dáta zapisujú do pomocnej premennej. Po nahromadení viac ako 5MB dát sa dáta zapíšu do súboru. Taktiež sa v tejto funkcii kontroluje sekvenčné číslo prijatého paketu, aby nedošlo k presakovaniu prijatých paketov. V prípade chyby pri kontrole sekvenčného čísla sa volá funkcia `transfer_error()`,

`transfer_error()`

Ukončuje spojenie a maže záznam z aktívnych spojení. Vypíše sa chybové hlásenie o neúspešnom prenose daného súboru.

`file_transferd()`

Slúži na korektné ukončenie spojenia a vypísanie hlásenia o úspešnom prijatí súboru. Zapisuje posledné dáta, ktoré sa akumulovali v dočasnej premennej pre obmedzenie I/O volaní a zatvára súbor.

`exit_server()`

Ukončuje beh serveru, uvoľňuje alokované zdroje a vypíše hlásenie o neukončených a prerušených spojeniach, ktoré neboli dokončené.

3 Preklad a spustenie programu

Program sa prekladá pomocou príkazu `make`, ktorý vytvorí spustiteľný súbor v koreňovom priečinku. Príkaz `make clean` vymaže dočasné súbory a vytvorené spustiteľné súbory.

Možné príklady spustenia:

```
$ ./secret -r movie.avi -s google.com
$ ./secret -r readme.md -s 127.0.0.1
$ ./secret -r readme.md -s 2001:4860:4860::8888
$ ./secret -r ./new_folder -s 127.0.0.1
$ ./secret -r movie.avi -s google.com -l
$ ./secret -l
$ ./secret -h
```

4 Výstup a ukončenie programu

Program vypisuje hlásenia o aktuálnom stave programu na štandardný výstup. Ukážky výpisu programu je možné nájsť v sekcii Testovanie. Viz sekcia 5.

Program by sa v žiadnom prípade nemal ukončiť neočakávanou chybou (`segmentation fault` a podobne). Možné chybové stavy sú chytené a ošetrené počas behu programu s príslušnými chybovými hláseniami. Pri akejkoľvek chybe sa vypíše chybové hlásenie na štandardný chybový výstup `stderr` a program skončí s chybovou návratovou hodnotou 1. Okrem toho sa v programe odchytáva aj signál `Keyboard interrupt`, po ktorom sa program korektne ukončí.

5 Testovanie

Použité stroje na testovanie správnosti programu boli:

- `PDS-VM (Referenčný virtualný stroj)`
- `Ubuntu 20.04.2.0 LTS`
- `Fedora release 33`
- `Manjaro 21.1.6`

Testovanie bolo zamerané na overenie funkčného posielania súboru bez straty akýchkoľvek dát. Taktiež sa testovala schopnosť odoslania viacerých súborov zároveň a prijatia viacerých súborov v tom istom čase. Pre overenie správnosti a integrity súboru bol použití program `md5sum` ktorý vygeneroval hash pre posielaný súbor a prijatý súbor, tieto dva vygenerované hash reťazce sa musia rovnať.

5.1 Test 1 (VM → Ubuntu)

Prenos súboru z virtuálneho stroja na Ubuntu po lokálnej sieti.

```
root@student-vm:# sudo ./secret -r ./to_send/2.jpg -s 172.21.62.22
File Name: ./to_send/2.jpg | Adress: 172.21.62.22
------------------------------
Sending file ... | Transfer ID: 151829943

Succesfully sended file: 2.jpg
```

Obr. 3: Klient poslal súbor `2.jpg` na server

```
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA/transfer$ sudo ../secret -l
----------------------------
Choosen interface for listen: eth0
Sever is listening!
----------------------------
Incoming file: 2.jpg | Transfer ID: 151829943
Succesfully transfered file: 2.jpg  | Transfer ID: 151829943
```

Obr. 4: Server prijal súbor `2.jpg` od klienta

```
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA$ md5sum ./transfer/2.jpg
f1ac80cedc670d797e7d84d23723f046  ./transfer/2.jpg
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA$ md5sum ./to_send/2.jpg
f1ac80cedc670d797e7d84d23723f046  ./to_send/2.jpg
```

Obr. 5: Kontrola hashu odoslaného a prijatého súboru

5.2 Test 2 (VM → Ubuntu)

Prenos viacerých súborov zároveň z VM na Ubuntu po lokálnej sieti.

```
root@student-vm:# sudo ./secret -r ./to_send/movie.avi -s 172.30.156.175
File Name: ./to_send/movie.avi | Adress: 172.30.156.175
-----------------------------
Sending file ... | Transfer ID: 38919889

Succesfully sended file: movie.avi
```

Obr. 6: Klient poslal súbor `movie.avi` na server

```
root@student-vm:# sudo ./secret -r ./to_send/ISA_1.mp4 -s 172.30.156.175
File Name: ./to_send/ISA_1.mp4 | Adress: 172.30.156.175
-----------------------------
Sending file ... | Transfer ID: 345954814

Succesfully sended file: ISA_1.mp4
```

Obr. 7: Klient poslal súbor `ISA_1.mp4` na server

```
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA/transfer$ sudo ../secret -l
----------------------------
Choosen interface for listen: eth0
Sever is listening!
----------------------------
Incoming file: ISA_1.mp4 | Transfer ID: 345954814
Incoming file: movie.avi | Transfer ID: 38919889
Succesfully transfered file: ISA_1.mp4  | Transfer ID: 345954814
Succesfully transfered file: movie.avi  | Transfer ID: 38919889
```

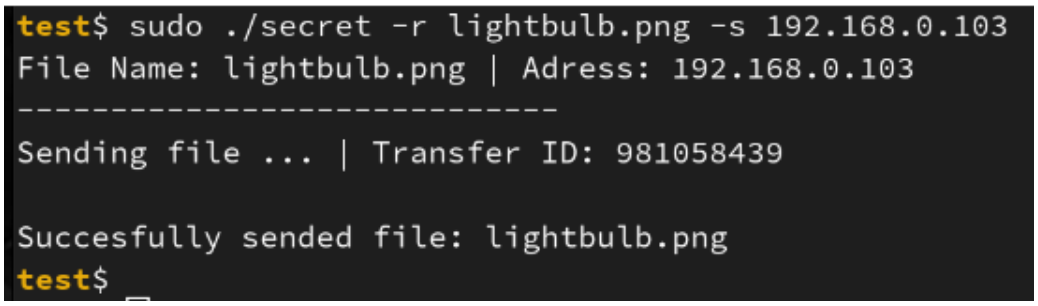
Obr. 8: Server prijal súbory `movie.avi` a `ISA_1.mp4` od klienta

```
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA$ md5sum ./to_send/movie.avi
bb1d4f7daafbe0a9ac2f2f24fef21c19  ./to_send/movie.avi
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA$ md5sum ./transfer/movie.avi
bb1d4f7daafbe0a9ac2f2f24fef21c19  ./transfer/movie.avi
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA$ md5sum ./to_send/ISA_1.mp4
95d4a3222bfa58cec469d2a9c3cbd214  ./to_send/ISA_1.mp4
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA$ md5sum ./transfer/ISA_1.mp4
95d4a3222bfa58cec469d2a9c3cbd214  ./transfer/ISA_1.mp4
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA$
```

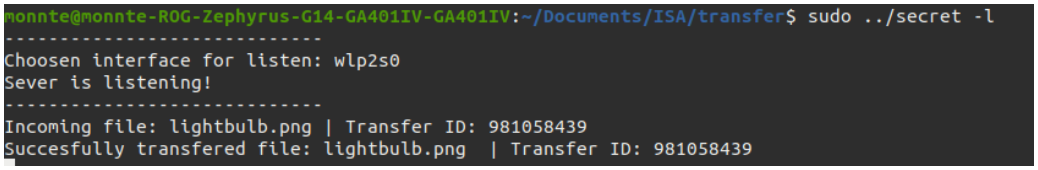
Obr. 9: Kontrola hashu odoslaných a prijatých súborov

5.3 Test 3 (Fedora → Ubuntu)

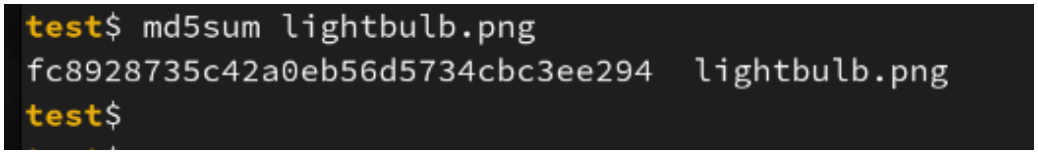
Prenos súboru z jedného počítača na druhý v lokálnej sieti.



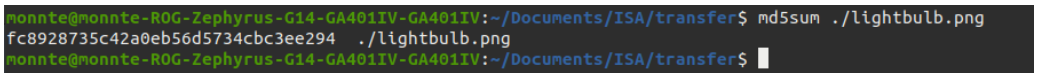
Obr. 10: Klient poslal súbor `lightbulb.png` na server



Obr. 11: Server prijal súbor `lightbulb.png` od klienta



Obr. 12: Kontrola hashu odoslaného súboru



Obr. 13: Kontrola hashu prijatého súboru

5.4 Test 4 (VM → Ubuntu)

Prenos priečinka z virtuálneho stroja na Ubuntu po lokalenj sieti.

```
root@student-vm:# sudo ./secret -r ./to_send/ -s 172.30.156.175
File Name: ./to_send/2.jpg | Adress: 172.30.156.175
-----------------------------
Sending file ... | Transfer ID: 45483232

Succesfully sended file: 2.jpg

File Name: ./to_send/ISA_1.mp4 | Adress: 172.30.156.175
-----------------------------
Sending file ... | Transfer ID: 45483232

Succesfully sended file: ISA_1.mp4

File Name: ./to_send/movie.avi | Adress: 172.30.156.175
-----------------------------
Sending file ... | Transfer ID: 45483232

Succesfully sended file: movie.avi

File Name: ./to_send/movie.srt | Adress: 172.30.156.175
-----------------------------
Sending file ... | Transfer ID: 45483232

Succesfully sended file: movie.srt

File Name: ./to_send/secret | Adress: 172.30.156.175
-----------------------------
Sending file ... | Transfer ID: 45483232

Succesfully sended file: secret
```

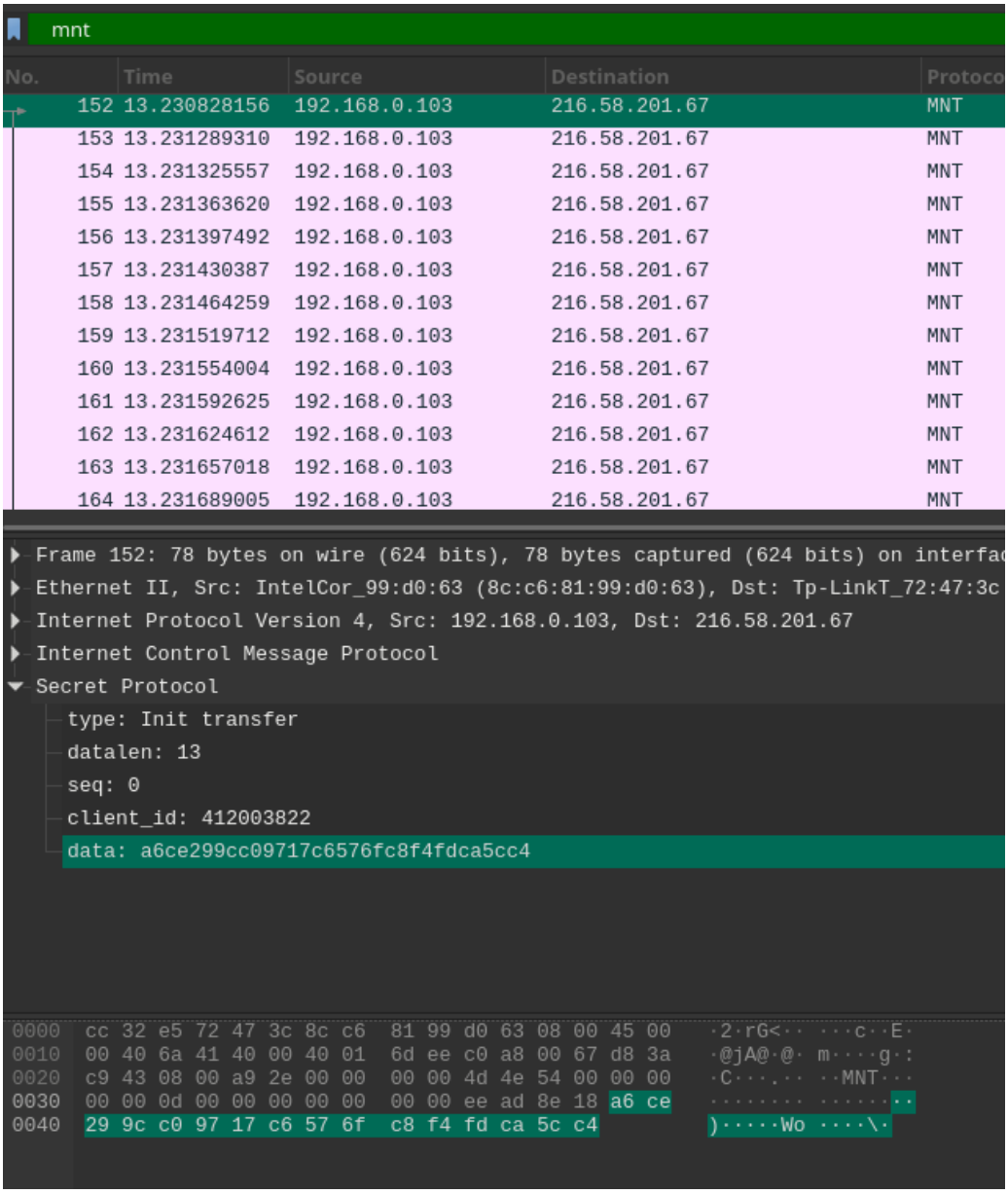
Obr. 14: Klient poslal priečinok `./to_send` na server

```
monnte@LAPTOP-RSMEGSQ2:/mnt/c/Users/zdrav/Desktop/Skola/3BIT_ZIM/ISA/ISA/transfer$ sudo ../secret -l
----------------------------
Choosen interface for listen: eth0
Sever is listening!
----------------------------
Incoming file: 2.jpg | Transfer ID: 45483232
Succesfully transfered file: 2.jpg  | Transfer ID: 45483232
Incoming file: ISA_1.mp4 | Transfer ID: 45483232
Succesfully transfered file: ISA_1.mp4  | Transfer ID: 45483232
Incoming file: movie.avi | Transfer ID: 45483232
Succesfully transfered file: movie.avi  | Transfer ID: 45483232
Incoming file: movie.srt | Transfer ID: 45483232
Succesfully transfered file: movie.srt  | Transfer ID: 45483232
Incoming file: secret | Transfer ID: 45483232
Succesfully transfered file: secret  | Transfer ID: 45483232
```

Obr. 15: Server prijal súbory z priečinka `./to_send` od klienta

5.5 Test 5 (Dissector)

Testovanie `dissectoru` v programe `wireshark`



5.6 Záver testovania

Z dôkladného testovania je možné usúdiť, že program je plne funkčný a dokáže preniesť a prijať súbor bez toho, aby sa poškodil.

Záver

Projekt spĺňa požiadavky zadania, nad rámec požiadaviek je implementovaná podpora prijímania viacerých súborov zároveň a zadanie názvu priečinka ako parameter pre poslanie súborov z klienta. Pre vymyslený protokol je vytvorený `dissector` do programu `Whireshark` pre jednoduchú identifikáciu paketov patriacich k tejto aplikácii.

Literatúra

[1] *Internet Control Message Protocol* [online]. RFC 792, [cit. 13.10.2021]. Dostupné z: https://rfc-editor.org/rfc/rfc792.txt, doi:10.17487/RFC0792.

[2] The Tcpdump Group, *Man page of PCAP* [online]. Posledná modifikácia: 9.9.2020 [cit. 13.10.2021]. Dostupné z: https://www.tcpdump.org/manpages/pcap.3pcap.html.

[3] *Parsing program options using getopt* [online]. [cit. 13.10.2021]. Dostupné z: https://www.gnu.org/software/libc/manual/html_node/Getopt.html.

[4] Bernstein, C., Cobb, M. Advanced Encryption Standard (AES) [online]. Posledná modifikácia: 7.2021 [cit. 13.10.2021]. Dostupné z: https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard.