

Síťové aplikace a správa sítí

Reverse-engineering neznámého protokolu

Tereza Burianová (xburia28)

Obsah

1	Protokol	2
1.1	Transmission Control Protocol (TCP)	2
1.2	Autentizace uživatele	2
1.3	Požadavky klienta	2
1.4	Odpovědi serveru	3
2	Použití	4
2.1	Dissector	4
2.2	Klient	4
3	Detaily implementace	4
3.1	Dissector	4
3.2	Klient	6
4	Testování klienta	6

Seznam obrázků

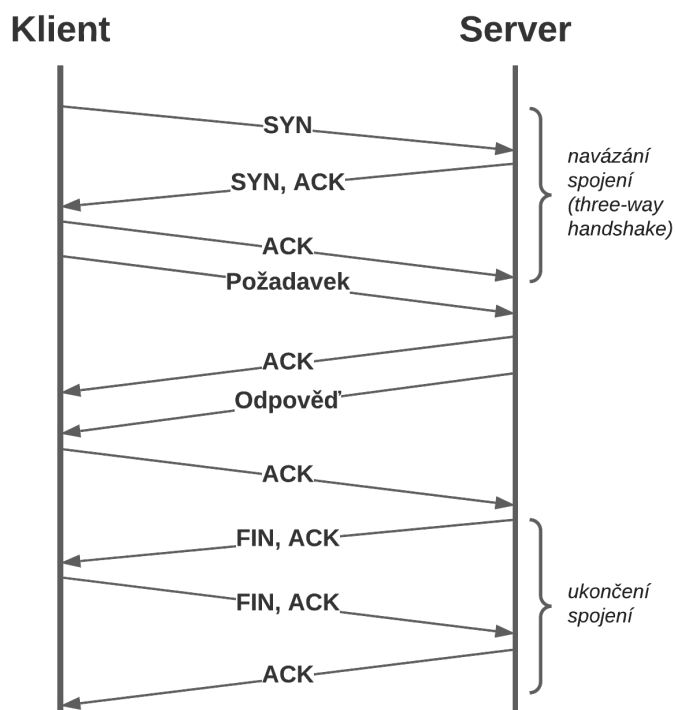
1	Komunikace mezi klientem a serverem	2
2	Příklad zobrazení požadavků dissectorem	4
3	Příklad zobrazení odpovědí dissectorem	4
4	Příklad zobrazení seznamu zpráv dissectorem	5
5	Stručné informace ve sloupci „Info“	5
6	Pakety spojené do jednoho v případě dlouhé zprávy (reassembly)	5
7	Porovnání odeslané a přijaté zprávy o velikosti 10 kB	6
8	Příkaz "list": vlevo výstup referenčního klienta, vpravo výstup implementovaného klienta.	7
9	Příkaz "fetch": Výstup referenčního klienta.	8
10	Příkaz "fetch": Výstup implementovaného klienta.	8

1 Protokol

Analyzovaný protokol umožňuje vytvoření několika uživatelů a následné odesílání a zobrazování zpráv. Jedná se tedy o jednoduchý komunikační nástroj.

1.1 Transmission Control Protocol (TCP)

Na základě odchycené komunikace bylo programem Wireshark zjištěno, že se jedná o TCP komunikaci. Před odesláním každého požadavku je tedy navázán three-way handshake (příznaky SYN, ACK) a po přijetí odpovědi je komunikace ukončena (příznaky FIN, ACK). TCP zaručuje spolehlivost a správné pořadí doručovaných informací.



Obrázek 1: Komunikace mezi klientem a serverem

1.2 Autentizace uživatele

Autentizace aktuálně přihlášeného uživatele je zajištěna pomocí tokenu uloženého v dočasném souboru login-token. Soubor je vytvořen pomocí **login** a smazán pomocí **logout**.

1.3 Požadavky klienta

Mezi požadavky, které mohou být na server odeslané, patří **register** (zaregistrování uživatele), **login** (přihlášení uživatele), **list** (zobrazení seznamu přijatých zpráv), **send** (odeslání zprávy), **fetch**

(zobrazení jednotlivé zprávy) a **logout** (odhlášení uživatele). Při odeslání jiného požadavku server odpoví chybovou hláškou „*unknown command*“.

Pro **register** a **login** jsou data odeslaná ve tvaru (**register/login** "login" "heslo"). V tento moment je již heslo zakódováno klientem pomocí kódování Base64.

Požadavky **list** a **logout** nevyžadují žádný argument, ale vyžadují odeslání tokenu sloužícího k autentizaci uživatele. Jsou tedy odeslány ve tvaru (**list/logout** "token").

V případě **fetch** je třeba ID zprávy, která se zobrazí. Je odeslán ve tvaru (**fetch** "token" id), kde ID je uvedeno, na rozdíl od ostatních argumentů, bez uvozovek.

Nejsložitějším požadavkem ze všech je **send**, musí obsahovat autentizační token, příjemce zprávy, předmět a samotný obsah. Je odeslán ve tvaru (**send** "token" "příjemce" "předmět" "zpráva").

1.4 Odpovědi serveru

Server může na přijatý požadavek odpovědět dvěma způsoby: byl vyřízen úspěšně (**ok**) nebo neúspěšně (**err**). Odpověď dále často obsahuje zprávu s dalšími informacemi o aktuálně provedené akci či chybě. Požadavek **login** vrací navíc i token přihlášeného uživatele, který bude využit pro autentizaci. Odpověď může vypadat například následovně:

(ok "user logged in" "c2VuZGVyMTYzNjNkxNDI5MzU1OC4xMzYy").

Speciální jsou odpovědi serveru na **list** a **fetch**. V případě **list** je zobrazen seznam zpráv obsahující jejich ID, odesílatele a příjemce. Každá zpráva je obsažena v kulatých závorkách. Konkrétně: (ok ((1 "sender" "subject1") (2 "sender" "subject2") (3 "sender" "subject3")))) a v případě prázdného seznamu: (ok ()).

Druhý z požadavků, **fetch**, zobrazuje konkrétní zprávu dle zadaného ID. Obsahuje odesílatele, předmět a samotný obsah zprávy. Tvar je následující: (ok ("sender" "subject1" "message")).

2 Použití

2.1 Dissector

Pro použití Lua dissectoru v programu Wireshark je třeba soubor **isa.lua** zkopírovat do jedné ze složek „*Personal (Lua) plugins*“ nebo „*Global (Lua) plugins*“. Jejich cestu je možno najít v nabídce *Help > About Wireshark > Folders*. Následně bude zpracovávat pakety na **portu 32323**, což je výchozí port použit vždy, když při spuštění serveru a klienta není stanoveno jinak.

2.2 Klient

Po spuštění souboru Makefile příkazem **make** je vytvořen spustitelný soubor **client**. Správný způsob použití lze zjistit pomocí příkazu `./client --help`, který pro program vypíše nápovědu.

3 Detaily implementace

3.1 Dissector

Dissector je implementován v jazyce Lua. V kódu jsou převzaty pouze krátké, většinou jednořádkové, úseky z fór, které jsou označeny komentáři s odkazy.

Pro zjednodušení orientace v zobrazené komunikaci je vytvořen samostatný „strom“ s názvem **ISA Protocol Data**, který vždy obsahuje typ paketu (**Message type**) a aktuálně zpracovávaný požadavek (**Request**). Odpověď vždy obsahuje informaci o úspěchu či neúspěchu (**Server response**). Dále jsou zobrazena všechna další data, která jsou z daného paketu zjistitelná, jako například uživatel a heslo v případě **register** a **login** požadavků nebo token, příjemce, předmět a obsah zprávy v případě **send** požadavku. U odpovědi je zobrazena i doplňující zpráva, pokud je dostupná.

ISA Protocol Data	ISA Protocol Data
Message type: request	Message type: request
Request: register	Request: fetch
User: sender	Login token: dGVyaTE2MzY5MTQyOTgwODAuNTQxNQ==
Encrypted password: dGVzdHBhc3M=	Message ID: 7

Obrázek 2: Příklad zobrazení požadavků dissectorem

ISA Protocol Data	ISA Protocol Data
Message type: response	Message type: response
Request: register	Request: register
Server response: ok	Server response: error
Message: registered user sender	Message: user already registered

Obrázek 3: Příklad zobrazení odpovědí dissectorem

V případě odpovědi na požadavek **list** jsou zobrazeny údaje o každé zprávě ve vlastním podstromu (konkrétně ID zprávy, odesílatel a předmět).

```

  ▾ ISA Protocol Data
    Message type: response
    Request: list
    Server response: ok
    Message count: 15
    ▾ Message 1
      Sender: sender
      Subject: subject1
    ▾ Message 2
      Sender: sender
      Subject: subject2
    ▾ Message 3
      Sender: sender
      Subject: subject3
    ▾ Message 4
      Sender: sender
      Subject: subject4

```

Obrázek 4: Příklad zobrazení seznamu zpráv dissectorem

Ve sloupci **Info** se zobrazuje stručná verze informací o paketu.

```

ISAMAIL 122 register request - user sender
ISAMAIL 117 register response - OK, registered user sender
ISAMAIL 118 register request - user sender
ISAMAIL 119 register response - ERR, user already registered
ISAMAIL 117 login request - user teri
ISAMAIL 114 login response - ERR, incorrect password
ISAMAIL 119 login request - user sender
ISAMAIL 144 login response - OK, token c2VuZGVyMTYzNjkxNDI5MzU1OC4xMzYy
ISAMAIL 129 list request - token c2VuZGVyMTYzNjkxNDI5MzU1OC4xMzYy
ISAMAIL 95 list response - OK, 0 message(s) found
ISAMAIL 157 send request - token c2VuZGVyMTYzNjkxNDI5MzU1OC4xMzYy, recipient teri, subject subject1
ISAMAIL 107 send response - OK, message sent

```

Obrázek 5: Stručné informace ve sloupci „Info“

V případě zprávy příliš dlouhé pro odeslání v jednom paketu se provede tzv. „reassembly“, které jednotlivé pakety patřící k sobě spojí.

```

  ▾ [5 Reassembled TCP Segments (10320 bytes): #396(4096), #398(222), #400(4096), #4
    [Frame: 396, payload: 0-4095 (4096 bytes)]
    [Frame: 398, payload: 4096-4317 (222 bytes)]
    [Frame: 400, payload: 4318-8413 (4096 bytes)]
    [Frame: 402, payload: 8414-8639 (226 bytes)]
    [Frame: 404, payload: 8640-10319 (1680 bytes)]
    [Segment count: 5]
    [Reassembled TCP length: 10320]
    [Reassembled TCP Data: 286f6b20282274657269222027375626a65637431352220224c6f7
  ▾ ISA Protocol Data
    Message type: response
    Request: fetch
    Server response: ok
    Sender: teri
    Subject: subject15
    Message body [truncated]: Lorem ipsum dolor sit amet, consectetur adipiscing

```

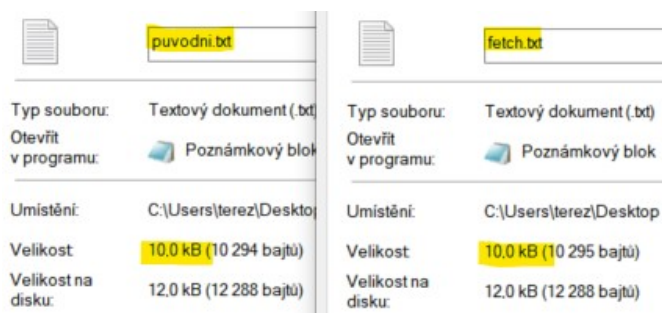
Obrázek 6: Pakety spojené do jednoho v případě dlouhé zprávy (reassembly)

3.2 Klient

Klient je implementován v jazyce C++ a je plně kompatibilní s referenčním klientem. Části kódu implementujícího prvotní nastavení a připojení byly převzaty z Beej's Guide to Network Programming [1] a následně upraveny pro použití v tomto projektu. Jedná se o jednotlivé funkce ze sítové knihovny BSD sockets, jejichž použití zdroj vysvětluje. Dále byl převzat kód v souboru *base64.h* [2], který implementuje zakódování pomocí Base64 potřebné pro hesla odesílaná na server. Všechny tyto úseky jsou v souborech s kódy označeny dle licenčních podmínek.

4 Testování klienta

Klient zvládá argumenty obsahující speciální znaky, prázdné argumenty i velmi dlouhé texty. Nejdelší text, který byl v rámci testování odeslán požadavkem **send**, měl velikost 10 kB. Jeho odeslání proběhlo v pořádku a přijatá zpráva, vypsaná požadavkem **fetch**, měla stejný obsah a velikost.



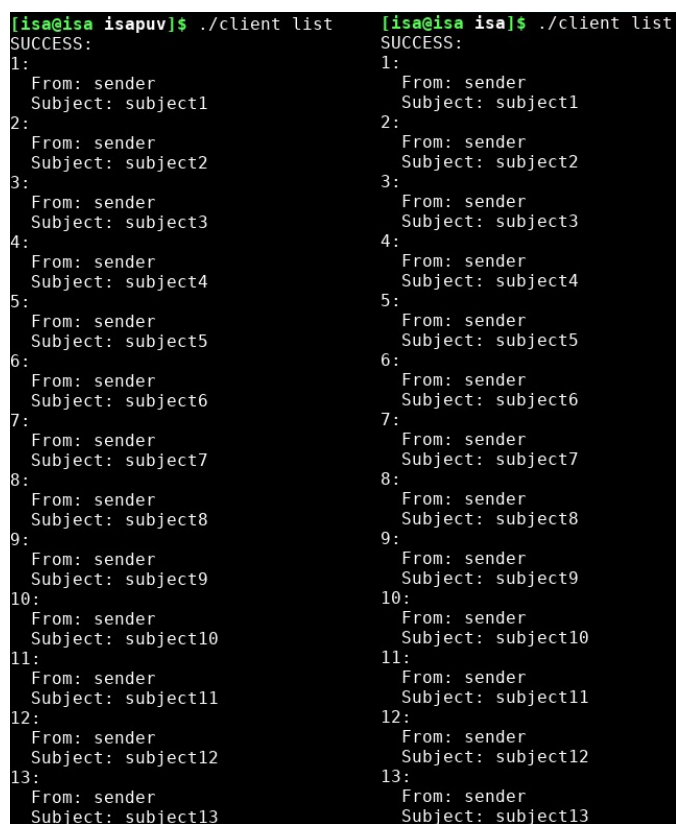
Typ souboru:	Textový dokument (.txt)	Typ souboru:	Textový dokument (.txt)
Otevřít v programu:	Poznámkový blok	Otevřít v programu:	Poznámkový blok
Umístění:	C:\Users\terez\Desktop	Umístění:	C:\Users\terez\Desktop
Velikost:	10.0 kB (10 294 bajtů)	Velikost:	10.0 kB (10 295 bajtů)
Velikost na disku:	12.0 kB (12 288 bajtů)	Velikost na disku:	12.0 kB (12 288 bajtů)

Obrázek 7: Porovnání odeslané a přijaté zprávy o velikosti 10 kB

K testování funkčnosti klienta, správnosti zpracování escape sekvencí a parsování výstupních dat serveru byla vytvořena následující testovací data:

```
./client send teri subject1 mes\sage
./client send teri subject2 mes\"sage
./client send teri subject3 mes\nsage
./client send teri subject4 "this is a test mes
sage"
./client send teri subject5 "this is a test mes'sage"
./client send teri subject6 "this is a test mes\sage"
./client send teri subject7 "this is a test mes\"sage"
./client send teri subject8 "this is a test mes\nsage"
./client send teri subject9 'this is a test mes
sage'
./client send teri subject10 'this is a test mes"sage'
./client send teri subject11 'this is a test mes\sage'
./client send teri subject12 'this is a test mes\"sage'
./client send teri subject13 'this is a test mes\nsage'
./client send teri subject15 'Lorem ipsum dolor sit amet [zkráceno - 10 kB textu]
./client send teri "" ""
```

Následně byly porovnány výsledky volání `./client list` a `./client fetch <id>` (poslední dva požadavky nejsou zobrazeny):



```
[isa@isa isapuv]$ ./client list
SUCCESS:
1:
  From: sender
  Subject: subject1
2:
  From: sender
  Subject: subject2
3:
  From: sender
  Subject: subject3
4:
  From: sender
  Subject: subject4
5:
  From: sender
  Subject: subject5
6:
  From: sender
  Subject: subject6
7:
  From: sender
  Subject: subject7
8:
  From: sender
  Subject: subject8
9:
  From: sender
  Subject: subject9
10:
  From: sender
  Subject: subject10
11:
  From: sender
  Subject: subject11
12:
  From: sender
  Subject: subject12
13:
  From: sender
  Subject: subject13

[isa@isa isa]$ ./client list
SUCCESS:
1:
  From: sender
  Subject: subject1
2:
  From: sender
  Subject: subject2
3:
  From: sender
  Subject: subject3
4:
  From: sender
  Subject: subject4
5:
  From: sender
  Subject: subject5
6:
  From: sender
  Subject: subject6
7:
  From: sender
  Subject: subject7
8:
  From: sender
  Subject: subject8
9:
  From: sender
  Subject: subject9
10:
  From: sender
  Subject: subject10
11:
  From: sender
  Subject: subject11
12:
  From: sender
  Subject: subject12
13:
  From: sender
  Subject: subject13
```

Obrázek 8: Příkaz "list": vlevo výstup referenčního klienta, vpravo výstup implementovaného klienta.


```

[isa@isa isapuv]$ ./client fetch 1
SUCCESS:
From: sender
Subject: subject1
message[isa@isa isapuv]$ ./client fetch 2
SUCCESS:
From: sender
Subject: subject2
mes"sage[isa@isa isapuv]$ ./client fetch 3
SUCCESS:
From: sender
Subject: subject3
mesnsage[isa@isa isapuv]$ ./client fetch 4
SUCCESS:
From: sender
Subject: subject4
this is a test mes
sage[isa@isa isapuv]$ ./client fetch 5
SUCCESS:
From: sender
Subject: subject5
this is a test mes'sage[isa@isa isapuv]$ ./client fetch 6
SUCCESS:
From: sender
Subject: subject6
this is a test mes\sage[isa@isa isapuv]$ ./client fetch 7
SUCCESS:
From: sender
Subject: subject7
[isa@isa isapuv]$ ./client fetch 8
SUCCESS:
From: sender
Subject: subject8
this is a test mes\nsage[isa@isa isapuv]$ ./client fetch 9
SUCCESS:
From: sender
Subject: subject9
this is a test mes
sage[isa@isa isapuv]$ ./client fetch 10
SUCCESS:
From: sender
Subject: subject10
this is a test mes"sage[isa@isa isapuv]$ ./client fetch 11
SUCCESS:
From: sender
Subject: subject11
this is a test mes\sage[isa@isa isapuv]$ ./client fetch 12
SUCCESS:
From: sender
Subject: subject12
this is a test mes"\sage[isa@isa isapuv]$ ./client fetch 13
SUCCESS:
From: sender
Subject: subject13
this is a test mes\nsage[isa@isa isapuv]$ █

```

Obrázek 9: Příkaz "fetch": Výstup referenčního klienta.

```

[isa@isa isa]$ ./client fetch 1
SUCCESS:
From: sender
Subject: subject1
message"
[isa@isa isa]$ ./client fetch 2
SUCCESS:
From: sender
Subject: subject2
mes"sage"
[isa@isa isa]$ ./client fetch 3
SUCCESS:
From: sender
Subject: subject3
mesnsage"
[isa@isa isa]$ ./client fetch 4
SUCCESS:
From: sender
Subject: subject4
this is a test mes\nsage"
[isa@isa isa]$ ./client fetch 5
SUCCESS:
From: sender
Subject: subject5
this is a test mes'sage"
[isa@isa isa]$ ./client fetch 6
SUCCESS:
From: sender
Subject: subject6
this is a test mes\sage"
[isa@isa isa]$ ./client fetch 7
SUCCESS:
From: sender
Subject: subject7
this is a test mes"sage"
[isa@isa isa]$ ./client fetch 8
SUCCESS:
From: sender
Subject: subject8
this is a test mes\nsage"
[isa@isa isa]$ ./client fetch 9
SUCCESS:
From: sender
Subject: subject9
this is a test mes\nsage"
[isa@isa isa]$ ./client fetch 10
SUCCESS:
From: sender
Subject: subject10
this is a test mes"sage"
[isa@isa isa]$ ./client fetch 11
SUCCESS:
From: sender
Subject: subject11
this is a test mes\sage"
[isa@isa isa]$ ./client fetch 12
SUCCESS:
From: sender
Subject: subject12
this is a test mes"\sage"
[isa@isa isa]$ ./client fetch 13
SUCCESS:
From: sender
Subject: subject13
this is a test mes\nsage"

```

Obrázek 10: Příkaz "fetch": Výstup implementovaného klienta.

Reference

- [1] Hall, B.: Beej's Guide to Network Programming. Dostupné z: <https://beej.us/guide/bgnet/html/>
- [2] (tomykaira) Tomita, M.: Base64.h. Dostupné z: <https://gist.github.com/0x3f00/90edbec0c04616d0b8c21586762bf1ac>