

Vysoké učení technické v Brně
Fakulta informačních technologií



Síťové aplikace a správa sítí

Manual pro project:
“Reverse-engineering neznámého protokolu”

Autor: Kozhevnikov Dmitrii (xkozhe00)

Brno
15.11.2021

Obsah

Cíl projektu.....	3
Zachycení protokolové komunikace.....	3
Implementování Wireshark dissektoru.....	4
Implementování kompatibilního klienta.....	5
Závěr.....	6
Seznam literatury.....	7

Cíl projektu

Cílem tohoto projektu bylo vytvořit komunikující aplikaci podle konkrétní vybrané specifikace pomocí síťové knihovny BSD. Projekt byl vytvořen v jazyce C++. Projekt se skládá z několika částí:

- Zachycení protokolové komunikace
- Implementování Wireshark dissektoru
- Implementování kompatibilního klienta

Zachycení protokolové komunikace

K zadání projektu byl dan virtuální stroj s referenčním klientem a serverem. Bylo nutné vytvořit spojení mezi klientem a serverem a poté pomocí Wiresharku zachytit tuto komunikaci a analyzovat ji. Výsledek zachycené komunikace je v souboru *isa.pcap*.

Byl nám chycen neznámý protokol TCP, který obsahoval pakety, které měly nešifrované data jak při odesílání na server, tak při přijímání odpovědi z něho.

Nešifrované údaje jsou odesílány zpravidla ve formátu: (*příkaz ["data"...]*)

A odpověď ve formátu: (*odpověď "informace"*)

Druh zprávy závisí na příkazu, který pošle klient.

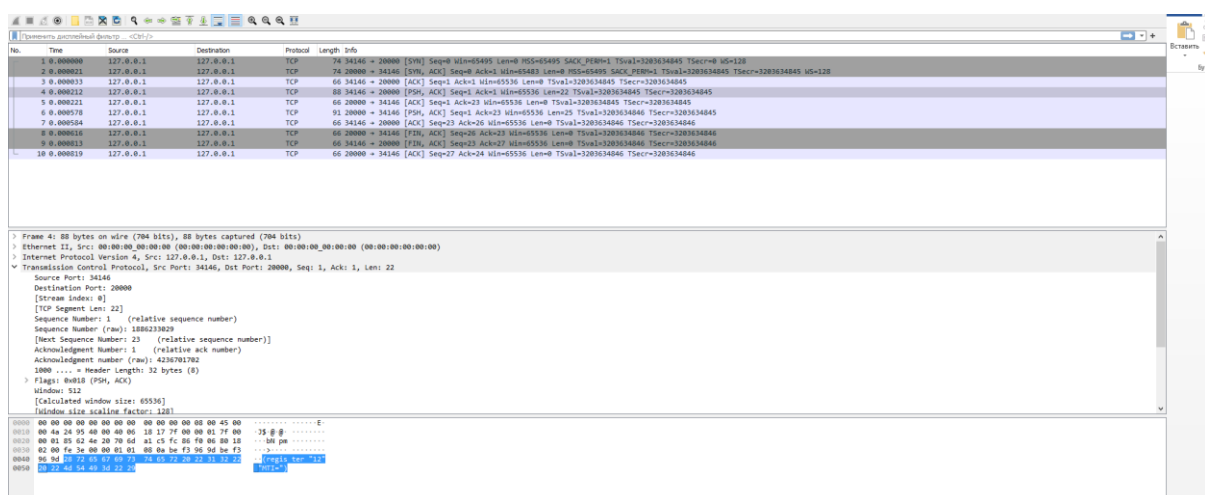
Příklad komandy *register*:

```
.....E·
·J$·@·@·
··bN pm
··>
··(register "12"
··"MTI=")
```

Client message

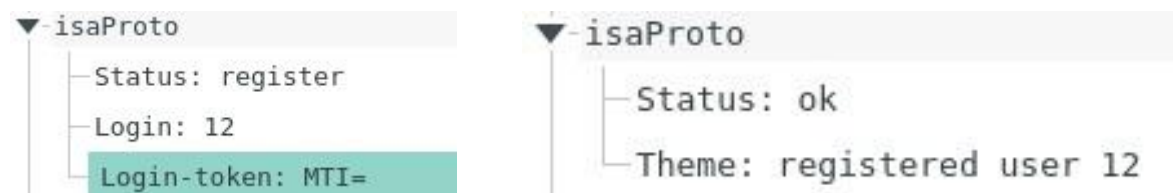
```
.....E·
·Mc{ @·@·
··N ·b· ·pm
··A·
··(ok "r egistere
d user 1 2")
```

Server response



Implementování Wireshark dissektoru

Pro podporu Wiresharku byl vytvořen vlastní disektor, který implementovan v jazyce Lua. Disektory umožňují prezentovat data protokolu v uživatelsky přívětivější podobě. Já jsem vytvořil disektor nazvaný *isaProto* pro neznámý protokol TCP, který poskytuje data o něm v uživatelsky přívětivém formátu.



Pro tento protokol tento disektor dává podrobné informace jak o požadavku klienta, tak o odpovědi ze serveru.

Capturing from Loopback: lo

No.	Time	Source	Destination	Protocol	Length	Info
18	11.533525829	127.0.0.1	127.0.0.1	TCP	66	20000 → 55902 [FIN, ACK] Seq=20 Ack=49 Win=65536 Len=0 TSval=4036916816 TSecr=4036916816
19	11.533775642	127.0.0.1	127.0.0.1	TCP	66	55902 → 20000 [FIN, ACK] Seq=49 Ack=21 Win=65536 Len=0 TSval=4036916817 TSecr=4036916816
20	11.533799214	127.0.0.1	127.0.0.1	TCP	66	20000 → 55902 [ACK] Seq=21 Ack=50 Win=65536 Len=0 TSval=4036916817 TSecr=4036916817
21	18.507243974	127.0.0.1	127.0.0.1	TCP	74	55904 → 20000 [SYN] Seq=0 Win=65495 Len=0 MSS=65495 SACK_PERM=1 TSval=4036923790 TSecr=0 WS=128
22	18.507272236	127.0.0.1	127.0.0.1	TCP	74	20000 → 55904 [SYN, ACK] Seq=0 Ack=1 Win=65483 Len=0 MSS=65495 SACK_PERM=1 TSval=4036923790 TSecr=4036923790 WS=128
23	18.507299387	127.0.0.1	127.0.0.1	TCP	66	55904 → 20000 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=4036923790 TSecr=4036923790
24	18.507695554	127.0.0.1	127.0.0.1	ISAPRO...	99	55904 → 20000 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=33 TSval=4036923790 TSecr=4036923790
25	18.507749840	127.0.0.1	127.0.0.1	TCP	66	20000 → 55904 [ACK] Seq=1 Ack=34 Win=65536 Len=0 TSval=4036923790 TSecr=4036923790
26	18.508086123	127.0.0.1	127.0.0.1	ISAPRO...	85	20000 → 55904 [PSH, ACK] Seq=1 Ack=34 Win=65536 Len=19 TSval=4036923791 TSecr=4036923790
27	18.508126948	127.0.0.1	127.0.0.1	TCP	66	55904 → 20000 [ACK] Seq=34 Ack=20 Win=65536 Len=0 TSval=4036923791 TSecr=4036923791
28	18.508264599	127.0.0.1	127.0.0.1	TCP	66	20000 → 55904 [FIN, ACK] Seq=20 Ack=34 Win=65536 Len=0 TSval=4036923791 TSecr=4036923791
29	18.509396752	127.0.0.1	127.0.0.1	TCP	66	55904 → 20000 [FIN, ACK] Seq=34 Ack=21 Win=65536 Len=0 TSval=4036923792 TSecr=4036923791
30	18.509416539	127.0.0.1	127.0.0.1	TCP	66	20000 → 55904 [ACK] Seq=21 Ack=35 Win=65536 Len=0 TSval=4036923792 TSecr=4036923792

Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1

Transmission Control Protocol, Src Port: 20000, Dst Port: 55904, Seq: 1, Ack: 34, Len: 19

isaProto

- Status: ok
- List number: 1
- From: 1
- Subject: 12

Data (19 bytes)

```

0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  :.....E..
0010  00 47 90 94 04 00 00 00 ac 2a 7f 00 00 01 7f 00   :G-Q@+...
0020  00 01 4e 20 da 60 2b b5 11 1f 08 60 26 57 80 10   :N'+...<...
0030  02 00 fe 3b 00 00 01 01 08 0a 0f 9e 91 8f 00 f0   :...~...<...
0040  91 9a 20 6f 6f 20 2b 28 31 20 22 31 20 22 31 20   :...ok (( 1 "1" *1
0050  32 22 29 29 29                                     :"))
    
```

[illegible]

Implementování kompatibilního klienta

Největší část projektu, která měla vytvořit klienta pro komunikaci se stávajícím serverem. Tento klient měl být co nejvíce podobný původnímu klientu. Nový klient byl vytvořen v jazyce C++. Tento klient má stejné funkce jako původní.

Také pro tohoto klienta je vytvořen **Makefile**, který provádí potřebné k sestavení, a který je třeba provést pomocí příkazu "make" před prvním spuštěním projektu.

Spuštění klienta lze provést z příkazového řádku pomocí příkazu:

`./myClient [<option> ...] <command> [<args>]...`

<option> je:

- `-a <addr>`, `--address <addr>` - název serveru nebo adresa, ke které se chcete připoji
- `-p <port>`, `--port <port>` - port serveru pro připojení k zadanému portu
- `-help`, `-h` - zobrazit tuto nápovědu

<command> je:

- `register <username> <password>` - registruje uživatele s daným jménem a heslem
- `login <username> <password>` - umožňuje přihlášení s daným přihlašovacím jménem a heslem pro uživatele je vytvořeno šifrované heslo v kódování **base64**, které je uloženo do souboru **token-login**.
- `list` - zobrazí seznam odeslaných zpráv
- `send <recipient> <subject> <body>` - odeslat zprávu pro příjemce, která má předmět zpravy a tělo
- `fetch <id>` - vrátí zprávu podle čísla
- `logout` - odhlásit se

```
[isa@isa isa]$ ./myClient -p 20000 -a 127.0.0.1 login 1 2
ERROR: incorrect password
[isa@isa isa]$ ./myClient -p 20000 -a 127.0.0.1 login 1 1
SUCCESS: user logged in
[isa@isa isa]$ ./myClient -p 20000 -a 127.0.0.1 send 1 12 123
SUCCESS: message sent
[isa@isa isa]$ ./myClient -p 20000 -a 127.0.0.1 list
SUCCESS:
1:
  From: 1
  Subject: 12
```

Závěr

Tento projekt mi umožnil rozšířit moje znalosti o pochopení fungování sítí. Lépe jsem pochopil, jak funguje Wireshark. Také byl prostudován materiál na zpracování paketu dat a práce dissektorů a možnost jejich implementace pro určité protokoly.

Také jsem byl schopen zvládnout implementace klienta, který rozšířil můj pohled na interakci “*client-server*“, zlepšil jsem znalostí práce s knihovnou BSD socket a naučil jsem se zpracovávat komunikaci.

Seznam literatury

1. <https://gitlab.com/wireshark/wireshark/-/wikis/Lua>
2. <https://gitlab.com/wireshark/wireshark/-/wikis/Lua/Dissectors>
3. <https://gitlab.com/wireshark/wireshark/-/wikis/Lua/Examples>
4. <https://en.wikipedia.org/>
5. https://en.wikiiversity.org/wiki/Internet_Protocol_Analysis/Internet_Layer_IPv4
6. <https://support.sas.com/documentation/onlinedoc/sasc/doc/lr2/lrv2ch15.htm>
7. <https://www.wireshark.org/>
8. <https://habr.com/>
9. <https://stackoverflow.com/questions/342409/how-do-i-base64-encode-decode-in-c>
10. <https://mika-s.github.io/wireshark/luadissector/2017/11/04/creating-a-wireshark-dissector-in-lua-1.html>
11. https://linux.die.net/man/3/getopt_long
12. Přednášky z předmětů IPK a ISA