

TCP/IPを模した3層アーキテクチャ

概要

本システムは、TCPを用いてTCP/IPのレイヤードアーキテクチャを模擬したシステムで、3層で構成される。第一層はデータ層、第二層はDTCPもしくはDUDP層、第三層はDIP層である。

第一層

第一層はデータ層である。

この層は、ヘッダとして下記の3フィールドを持つ。

フィールド	長さ(Bytes)	意味
type	4	上位プロトコルタイプを表す
version	4	DIPバージョンを示す
ttl	4	time to live

第二層

第二層は、DTCPもしくはDUDP用のデータ層である。

DTCP(protocol type 6)

この層は、ヘッダとして下記の3フィールドを持つ。ここで、ペイロードのMD5値が不正なパケットと判断された場合、改ざんされていた場合には即座に処理を終了する。

フィールド	長さ(Bytes)	意味
type	4	上位プロトコルタイプを表す
len	4	ペイロードの長さ
digest	4	ペイロードのMD5値

DUDP(protocol type 17)

この層は、ヘッダとして下記の2フィールドを持つ。

フィールド	長さ(Bytes)	意味
type	4	上位プロトコルタイプを表す
len	4	ペイロードの長さ

第一層

この層はデータを管理する。

必要なもの(Ubuntu)

- gcc
- make
- libcunit1-dev
- libc-dev

必要なもの(Docker)

- Docker
- make

使い方

サーバを起動してからクライアントを実行することで、クライアントからサーバにリクエストを飛ばすことができる。

ローカルでの実行

```
make build
./bin/server &
sleep 3 # サーバが立つまで少し待機
./bin/client 17 build.sh # dudpでbuild.shを送る
./bin/client 6 build.sh # dtcpでbuild.shを送る
```

Dockerコンテナ内での実行

```
make docker-build
make docker-run
# ここからコンテナ内
./binserver &
sleep 3 # サーバが立つまで少し待機
./bin/client 17 build.sh # dudpでbuild.shを送る
./bin/client 6 build.sh # dtcpでbuild.shを送る
```

実行例

DTCPを指定してクライアント(右)からサーバ(左)にbuild.shのデータを送信した例

```

user@user-VirtualBox:~/work/class$ ./bin/server
server waiting
[server]
=====DIP=====
type : 6
version: 1
ttl : 64
=====
type : 10
len : 126
digest : D6 7A 97 24 4D 0F 38 25 62 75 F9 C3 A4 FB 2D FE
actual : D6 7A 97 24 4D 0F 38 25 62 75 F9 C3 A4 FB 2D FE
=====
RAWDATA: #!/bin/sh

make build

kill -9 $(ps a | grep ./bin/server | awk '{print $1}' | head -1)

./bin/server &
sleep 3
./bin/client

server waiting

[END] send_payload
[END] client_call
user@user-VirtualBox:~/work/class$ clear
user@user-VirtualBox:~/work/class$ ./bin/client 6 build.sh
[BEGIN] client_call
[BEGIN] build_payload
#!/bin/sh

make build

kill -9 $(ps a | grep ./bin/server | awk '{print $1}' | head -1)

./bin/server &
sleep 3
./bin/client

payload:
06 00 00 00 01 00 00 00 40 00 00 00 0A 00 00 00
7E 00 00 00 D6 7A 97 24 4D 0F 38 25 62 75 F9 C3
A4 FB 2D FE 23 21 2F 62 69 6E 2F 73 68 0A 0A 6D
61 6B 65 20 62 75 69 6C 64 0A 0A 6B 69 6C 6C 20
2D 39 20 24 28 70 73 20 61 20 7C 20 67 72 65 70
20 2E 2F 62 69 6E 2F 73 65 72 76 65 72 20 7C 20
61 77 6B 20 27 7B 70 72 69 6E 74 20 24 31 7D 27
20 7C 20 68 65 61 64 20 2D 31 29 0A 0A 2E 2F 62
69 6E 2F 73 65 72 76 65 72 20 26 0A 73 6C 65 65
70 20 33 20 0A 2E 2F 62 69 6E 2F 63 6C 69 65 6E
74 0A
[END] build_payload
[BEGIN] send_payload
recv data:
06 00 00 00 01 00 00 00 40 00 00 00 0A 00 00 00
7E 00 00 00 D6 7A 97 24 4D 0F 38 25 62 75 F9 C3
A4 FB 2D FE 23 21 2F 62 69 6E 2F 73 68 0A 0A 6D
61 6B 65 20 62 75 69 6C 64 0A 0A 6B 69 6C 6C 20
2D 39 20 24 28 70 73 20 61 20 7C 20 67 72 65 70
20 2E 2F 62 69 6E 2F 73 65 72 76 65 72 20 7C 20
61 77 6B 20 27 7B 70 72 69 6E 74 20 24 31 7D 27
20 7C 20 68 65 61 64 20 2D 31 29 0A 0A 2E 2F 62
69 6E 2F 73 65 72 76 65 72 20 26 0A 73 6C 65 65
70 20 33 20 0A 2E 2F 62 69 6E 2F 63 6C 69 65 6E
74 0A
[END] send_payload
[END] client_call

```

DUDPを指定してクライアント(右)からサーバ(左)にbuild.shのデータを送信した例

```

user@user-VirtualBox:~/work/class$ ./bin/server
server waiting
[server]
=====DIP=====
type : 17
version: 1
ttl : 64
=====
type : 10
len : 126
RAWDATA: #!/bin/sh

make build

kill -9 $(ps a | grep ./bin/server | awk '{print $1}' | head -1)

./bin/server &
sleep 3
./bin/client

server waiting

70 20 33 20 0A 2E 2F 62 69 6E 2F 63 6C 69 65 6E
74 0A
[END] send_payload
[END] client_call
user@user-VirtualBox:~/work/class$ ./bin/client 17 build.sh
[BEGIN] client_call
[BEGIN] build_payload
payload:
11 00 00 00 01 00 00 00 40 00 00 00 0A 00 00 00
7E 00 00 00 23 21 2F 62 69 6E 2F 73 68 0A 0A 6D
61 6B 65 20 62 75 69 6C 64 0A 0A 6B 69 6C 6C 20
2D 39 20 24 28 70 73 20 61 20 7C 20 67 72 65 70
20 2E 2F 62 69 6E 2F 73 65 72 76 65 72 20 7C 20
61 77 6B 20 27 7B 70 72 69 6E 74 20 24 31 7D 27
20 7C 20 68 65 61 64 20 2D 31 29 0A 0A 2E 2F 62
69 6E 2F 73 65 72 76 65 72 20 26 0A 73 6C 65 65
70 20 33 20 0A 2E 2F 62 69 6E 2F 63 6C 69 65 6E
74 0A
[END] build_payload
[BEGIN] send_payload
recv data:
11 00 00 00 01 00 00 00 40 00 00 00 0A 00 00 00
7E 00 00 00 23 21 2F 62 69 6E 2F 73 68 0A 0A 6D
61 6B 65 20 62 75 69 6C 64 0A 0A 6B 69 6C 6C 20
2D 39 20 24 28 70 73 20 61 20 7C 20 67 72 65 70
20 2E 2F 62 69 6E 2F 73 65 72 76 65 72 20 7C 20
61 77 6B 20 27 7B 70 72 69 6E 74 20 24 31 7D 27
20 7C 20 68 65 61 64 20 2D 31 29 0A 0A 2E 2F 62
69 6E 2F 73 65 72 76 65 72 20 26 0A 73 6C 65 65
70 20 33 20 0A 2E 2F 62 69 6E 2F 63 6C 69 65 6E
74 0A
[END] send_payload
[END] client_call

```

テスト

テストツールとしてCUnitを採用しました。

```
make test
(snip)
***** CUNIT CONSOLE - MAIN MENU *****
(R)un (S)elect (L)ist (A)ctivate (F)ailures (O)ptions (H)elp (Q)uit
Enter command: R
(snip)
```

ライセンス関連

- [RSA Data Security, Inc. MD5 Message-Digest Algorithm](#)