Hacking with Linux networking cli tools

Xiaolin WANG (wx672ster+net@gmail.com)

2024-06-30

Caution

- You must submit your report as a `tar ball` in which the following files should be included:
  1. Your report in either `Emacs Org` or `Markdown` format, and a PDF file generated from your `org` or `md` file. Tips:
     - In Emacs, press `C-c C-e l p` to export PDF file from your org file;
     - For `Markdown` to PDF, you can try `markdown`, `pandoc`, `cmark`, whatever. For example:

       ```
       pandoc input.md --pdf-engine=lualatex -o output.pdf
       ```

     - This HTML page itself is generated from an markdown file (proj-week.md). You can take it as an example.
     - **Report template** `org file`, `html file`, `markdown file`
  2. your program source files (bash scripts, C programs).
  3. a `ttyrec` file recording your operations (`man ttyrec`).

Here' s how:

1. make a directory, e.g. 20231159xxx. In this directory, try very hard to make all the files available.

```
mkdir  20231159xxx       # create a new directory
cd 20231159xxx
vim tmux-http.sh         # write your script
vim tcpServer.c          # Implement the TCP server in C
vim tcpClient.c          # Implement the TCP client in C
vim 20231159xxx.org      # write your report with emacs-org, or
vim 20231159xxx.md       # write your report in markdown format
ttyrec http-demo.ttyrec # make your demo screencast
```

2. make a tar ball.

   ```
   cd ..
   tar zcf 20231159xxx.tgz 20231159xxx
   ls -l # make sure your tar ball is smaller than 1MB in size
   ```

3. upload the `tgz` file to our moodle site.

- Here is a short *video* tutorial on writing lab report: `tutorial.ttyrec`. To view it:

  ```
  ttyplay  tutorial.ttyrec
  ```

  Feel free to make your own `ttyrec` file while doing this lab work. For example:

  ```
  ttyrec  20231159xxx-http.ttyrec
  ttyrec  20231159xxx-email.ttyrec
  ttyrec  20231159xxx-ftp.ttyrec
  ```

- **Bonus points:** Manage your project with `git`. `man gittutorial` to learn the very basics of it.
- **Deadline:** <2024-7-7 Sun>
  - Submit your report as a `tgz` file here. In your `tgz` file, there must be:
    1. your report in `org` or `markdown` format
    2. your report in PDF format
    3. your bash script for demostrating a HTTP session
    4. one or more `ttyrec` files for demostrating whatever you did
  - Late reports will be penalized 20% per day.
  - MS-word file will **NOT** be accepted. Cheating will result in automatic failure of this work.

`tmux, nc, ip, tcpdump, ss, nmap, curl`

Here are the bash scripts I used in the class for demostrating how some protocols work.

- TCP three-way handshake
- UDP
- SMTP (need a SMTP server)
- FTP (need a FTP server)

- **Your tasks:**
  1. Run the above scripts to get familar with these tools, and get a thorough understanding about these protocols;

2. Packet analysis. Upon running the following command:

   ```
   sudo tcpdump -ilo -nnvvvxXKS -s0 port 3333
   ```

   the following packet is captured:

   ```
   08:34:10.790666 IP (tos 0x0, ttl 64, id 12824, offset 0, flags
   [DF], proto TCP (6), length 64)

   127.0.0.1.46668 > 127.0.0.1.3333: Flags [P.], seq
   2400005725:2400005737, ack 373279396, win 512, options
   [nop,nop,TS val 3259949783 ecr 3259896343], length 12

   0x0000:  4500 0040 3218 4000 4006 0a9e 7f00 0001  E..@2.@.@.......
   0x0010:  7f00 0001 b64c 0d05 8f0d 2e5d 163f caa4  .....L.....].?..
   0x0020:  8018 0200 fe34 0000 0101 080a c24e e2d7  .....4.......N..
   0x0030:  c24e 1217 6865 6c6c 6f20 776f 726c 640a  .N..hello.world.
   ```

3. Tell me the meaning of each option used in the previous command.
4. Please analyze this captured packet and explain it to me as detailed as you can.
5. Write a similar script showing how HTTP works (you need `curl`);
6. Record your HTTP demo session with `ttyrec`.

Socket programming

The followings are the Python programs I used in the class for demostrating socket programming. Your tasks:

1. Try these programs with a remote server IP instead of 127.0.0.1.
2. Rewrite them in C.

TCP

A simple TCP server written in Python3

```
#!/usr/bin/python3

### A simple TCP server ###

from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(0)
print(serverSocket.getsockname())
print('The server is ready to receive')
```

```
while 1:
    connectionSocket, addr = serverSocket.accept()
    print(connectionSocket.getsockname())
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

A simple TCP client written in Python3

```
#!/usr/bin/python3

### A simple TCP client ###

from time import *
from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
print(clientSocket.getsockname())
sentence = input('Input lowercase sentence:')
clientSocket.send(bytes(sentence,'utf-8'))
modifiedSentence = clientSocket.recv(1024)
print('From Server:', str(modifiedSentence,'utf-8'))
clientSocket.close()
```

A simple TCP demp script

```
#!/bin/bash

### A simple TCP demo script ###

set -euC

tmux rename-window "TCP demo"

#    Window setup
# +--------+--------+
# | server | client |
# +--------+--------+
# |      watch      |
# +-----------------+
# |     tcpdump     |
# +-----------------+
#
```

```
tmux split-window -h
tmux split-window -fl99
tmux split-window -l12

tmux send-keys -t{top-left} "./tcpServer.py"

tmux send-keys -t{top-right} "./tcpClient.py"

tmux send-keys -t{up-of} "watch -tn.1 'ss -ant \"( sport == 12000 or dport == 12000 )\"'" C-

tmux send-keys "sudo tcpdump -ilo -vvvnnxXSK -s0 port 12000" C-m
```

UDP

A simple UDP server written in Python3

```
#!/usr/bin/python3

### A simple UDP server ###

from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Input lowercase sentence:')
clientSocket.sendto(bytes(message,'utf-8'),(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(str(modifiedMessage,'utf-8'))
clientSocket.close()
```

A simple UDP client written in Python3

```
#!/usr/bin/python3

### A simple UDP client ###

from socket import *
serverName = '127.0.0.1'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = input('Input lowercase sentence:')
clientSocket.sendto(bytes(message,'utf-8'),(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print(str(modifiedMessage,'utf-8'))
clientSocket.close()
```

A simple UDP demp script

```
#!/bin/bash

### A simple UDP demo script ###

set -euC

tmux rename-window "UDP demo"

#    Window setup
# +--------+--------+
# | server | client |
# +--------+--------+
# |     tcpdump     |
# +-----------------+
#
tmux split-window -h
tmux split-window -fl99

tmux send-keys -t{top-left}  "./udpServer.py"
tmux send-keys -t{top-right} "./udpClient.py"

tmux send-keys "sudo tcpdump -ilo -vvvnnxXK port 12000" C-m
```