Network Administration/System Administration (NTU CSIE, Spring 2019) Homework #1

Network Administration

NANI? Is my uplink as evil as Q-mao? (15%)

1. Find the packet that contains the email and password you just typed, and screenshot it. (5%)

```
30    0.056568751 192.168.71.128    140.112.30.26     HTTP  1130  POST
/nasahw1-login?login_attempt=1&lwv=110 HTTP/1.1
(application/x-www-form-urlencoded)
```

```
0050  65 6d 61 69 6c 3d 6e 61  73 61 68 77 31 25 34 30   email=na sahw1%40
0060  71 6d 61 6f 2e 69 73 2e  73 6f 2e 65 76 69 6c 26   qmao.is. so.evil&
0070  70 61 73 73 3d 42 30 37  36 31 31 30 31 32 26 74   pass=B07 611012&t
0080  69 6d 65 7a 6f 6e 65 3d  2d 36 30 30 26 6c 67 6e   imezone= -600&lgn
0090  64 69 6d 3d 65 79 4a 33  49 6a 6f 78 4f 54 49 77   dim=eyJ3 IjoxOTIw
00a0  4c 43 4a 6f 49 6a 6f 78  4d 44 67 77 4c 43 4a 68   LCJoIjox MDgwLCJh
00b0  64 79 49 36 4d 54 67 31  4d 79 77 69 59 57 67 69   dyI6MTg1 MywiYWgi
00c0  4f 6a 45 77 4e 54 4d 73  49 6d 4d 69 4f 6a 49 30   OjEwNTMs ImMiOjI0
00d0  66 51 25 33 44 25 33 44  26 6c 67 6e 72 6e 64 3d   fQ%3D%3D &lgnrnd=
00e0  30 30 32 35 30 39 5f 37  50 77 78 26 6c 67 6e 6a   002509_7 Pwx&lgnj
00f0  73 3d 31 35 35 32 32 31  33 36 39 36 26 61 62 5f   s=155221 3696&ab_
0100  74 65 73 74 5f 64 61 74  61 3d 41 41 41 41 25 32   test_dat a=AAAA%2
0110  46 25 32 46 41 41 41 41  41 41 41 41 41 41 41 41   F%2FAAAA AAAAAAAA
```

2. Re-perform the above steps at fakebook-https2, can you find the email and password this time? If not, does it mean the owner of the web server cannot peek your password during your login? Why or why not? (5%)

只能看到一堆protocol標示為TCP或TLSv1.2的封包，更上層的內容都被加密了。但server端握有非對稱加密的私鑰，所以依舊可以解開密文。

```
▾ Extension: SessionTicket TLS (len=224)
    Type: SessionTicket TLS (35)
    Length: 224
    Data (224 bytes)
▾ Extension: application_layer_protocol_negotiation (len=14)
```

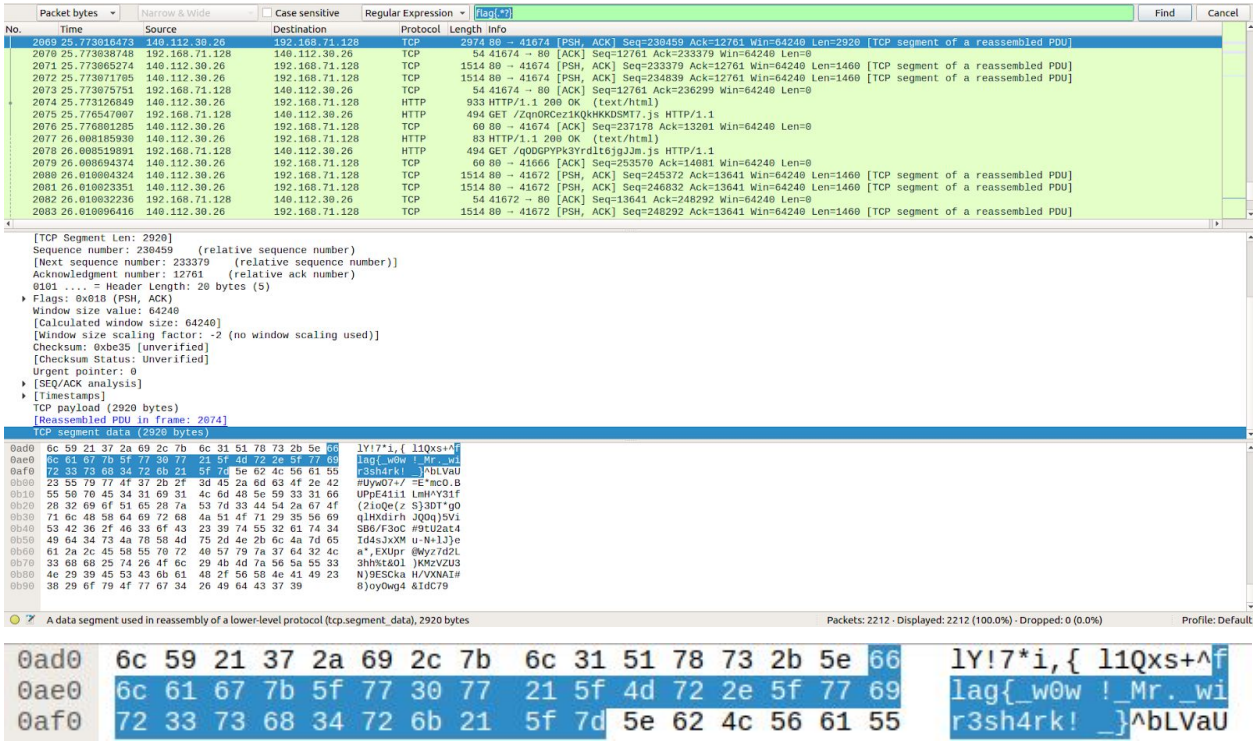
```
00f0  af 6b 90 db ff 38 9c bf  5d be 43 9e 1b 24 bd 9c   ·k···8·· ]·C··$··
0100  21 64 a4 35 95 eb 62 5b  77 41 c3 22 f5 cf c5 cf   !d·5··b[ wA·"····
0110  eb 7d 94 90 6a a5 d8 0a  5b cc d5 18 44 fb d1 5d   ·}··j··· [···D··]
0120  ac e2 66 69 d0 6b da 70  c2 07 41 ed 40 08 d4 73   ··fi·k·p ··A·@··s
0130  cb 20 47 38 cd bf 33 ef  8f f1 6c 48 89 83 e8 7e   · G8··3· ··lH···~
0140  29 90 3a 55 77 04 be f9  1e cf 92 7c 1c 87 a8 46   )·:Uw··· ···|···F
0150  ee c4 40 d0 99 cd 7b 53  65 d0 4d 79 eb 88 7d 4f   ··@···{S e·My··}O
0160  52 23 d3 b2 06 68 8d 34  f4 f3 5c 14 c3 f5 66 0f   R#···h·4 ··\···f·
0170  98 25 55 85 0d 47 7f 3e  66 4e 2a 1e 4c 87 fb d2   ·%U··G·> fN*·L···
0180  37 db 17 22 30 b0 c7 9c  cc ed 7d 0c cf 73 ef 55   7··"0··· ··}··s·U
0190  62 58 94 09 8d 86 8a 8b  95 76 46 b2 4b 71 a1 69   bX····· ·vF·Kq·i
01a0  25 46 8c 13 6c 74 b2 a9  c3 0d 51 3a c8 82 6c 1d   %F··lt·· ··Q:··l·
01b0  fb 07 e7 eb 26 2f a9 82  a7 d8 b3 43 e4 15 37 a4   ····&/·· ···C··7·
01c0  a2 78 1f 5a 76 1d ab 94  7f 4f 60 73 e8 d5 47 d3   ·x·Zv··· ·O`s··G·
```

3. Give a real-world examples that your password may be eavesdropped by Q-mao when still using the HTTP protocol. (assuming no eavesdropper in client and server-side software) (5%)
當你使用路上的免費WiFi時，如果使用的網站(或app)使用的傳輸協定是沒有加密的，提供WiFi的機器就可以側錄下你傳輸的內容。

In the vast sea, I see nobody but flag (10%)

```
C-f "Packet bytes" "Regular Expression" "flag{.*?}" // “?” for greedy
```

Reference: wireshark.org




```
flag{_w0w!_Mr._wir3sh4rk!_}
```

Piepie! The world needs you! (10%)

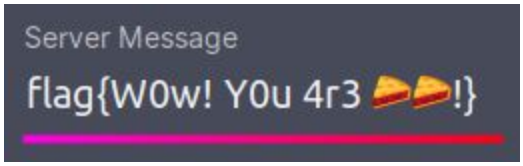
用wireshark打開history.pcapng，首先搜尋flag{.*?}，找到No.20472可是沒有flag，他告訴你要自己送一遍，對他右鍵Follow > TCP stream，可以看到request是哪一包，看到上一題給的token&answer。

```
POST /answer HTTP/1.1
Host: nasa-hw1.csie.ntu.edu.tw
User-Agent: python-requests/2.18.4
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
Content-Length: 2076
Content-Type: application/x-www-form-urlencoded

token=e03d2e04900b454661e46cdba510cfddd2b80b449e804113402201f053d085bf5495a4368eafc1afc68f00018505e68ff5e95e7d6f4f9d1d70919da3a7fc992
85b28469c2cfc711c56aa423f561179d2dd6f18ce2a820e242bce4744904086e4fcd3b8420135be9afa9374ef856b2b71bff5e07ca9debacfea25bd67cfd590297f05
0738fc73737a11b7883e4ed556f68d07994c8d386842f87ac53bd152c58d85d54dbeb4bfdc30cfb2a1c20f85f5d931c5cbc551ba5fb007aca75653d45ae79230793e7
5d833c1ab0a23c20f1537f59083736d623b0aaf5503d5453a5344d8b7bbc1400735093e15b8b62966236ab4c08569532a8b6558b861c147764db5e3fc01532d8a9488
abf44b45821e4c5fefddcf0581b30b1f5b4ffa3179fd15d62b628339813cbc9ab33dcd3a0c7c3f792075c5ddfdb9bb415fdb7cdfcdf41f959d3a790fd8353ae82ef3a
db45cb38a66c646f59d626dc18786b20214969f2ed7d9236b55f582760ce79e89906721100e70423fb6ba6254712f2c80c7abf40e1432b85017d3270e5cfb1915d831
294fbd6aa6a7f2f2c0a7c52b08d7c56f6a92b162ed5519a21597ada0ddedb649cfb16dd1217f49293ec7c2e4240f0c409c4250c47b2ce102045be1579edf36a38772d
6495f9190cbec44083b9b7e3412edd863e7bc60d023dc0972b008962ced619fd9b5c4a11baff4283c17d3e2ade4ab8235e5269b12a990556a9605f2c2e8e257f2f169
dab812e26dbc9785ee0d874f8dc66471e2db1ff65bd83f480945fc841169714b97ed988b0fe2af7be43eb775a91cad399a35e8b08da4eaa6cc96e5f0dab4774c9be9c
fdf1be7d83e6b8be20b2587f4306b8af6bd4beaa54141b7945d69dee33799b80ab967443856b4624c5a6caaf07ee0d002ae730da6653479f9d3098257fc4e2cce6c49
3ab795d2c3cdf6f007a161627eb06cef400afd77806d6567e851d879e8d749d56f318d4369cb7b0c59bdbcaf128c56c34f8bef41f0f541bb84ee676bd1f9afd748ea8
8e7e55e2033bc374cc49900cea5d97883ebcd827c060c244a5926746b8f41a9f40c0222ed883b5dcefbcfc852d92ee9deaefa10721032b89aa7e16e427af6e0664260
025674a83985d0d050ecdbc482b7410e8dc3eecf28d4c5759fdc0b5f17fd83bf5485661aac00cafff5ad7decaadf999c1b8113b6af8680d39ea03f631de4ae5173323
7a15b9646c8e8bc41992f30bcd39729dae91e6c1274e02a9fdeb007cd37fc962f28acd87748e863e94ca8dec2295b74e18f318fa178883ba9139e7fe914db76746c0b
ce9a3ff501edc4ee73955d8d26553ce9c55ea699086345346f03752d20c255a8a3eeb5b3e0b0f8d52693ff884335e1e977c7d5400ec85d8c1ce8db1381388da31f34d
2980c7ba81e8ea13c8ff446919581b938d4554b8d424455d6334cde85a6&answer=A+PIEce+of+PIEHTTP/1.1 200 OK
Date: Sun, 24 Feb 2019 16:27:52 GMT
Server: Werkzeug/0.14.1 Python/3.6.6
Content-Type: text/html; charset=utf-8
Content-Length: 231
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

{"content": "Welcome home, piepie! <br><br> https://www.youtube.com/watch?v=uAsV5-Hv-7U", "exit": 1, "msg": "flag{This is not flag,
you need to replay the corresponding answers in yourself to get your flag.}", "title": "Congrats!"}
```

接下來搜尋是哪個對話給了這個token，一路找下去可以整理出問題跟答案，再去一題一題回答，偶爾會出現不在這裡的問題，只好重來了

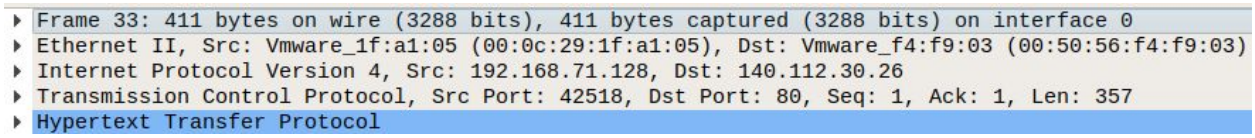
```
Q."Piepie or pie?"
A PIEce of PIE
Q."What is your favorite fruit?"
Pieapple
Q."What is the question?"
To pie or not to pie
Q."Who is your favorite pie maker?"
Don McLean
Q."How pie are you?"
PIEr than a pied pieman
```

"msg": "flag{W0w! Y0u 4r3 \ud83e\udd67\ud83e\udd67!}"



Internet Protocol Stack: 5-layer Model

1. Wireshark can reveal the data fields used by protocols of hierarchical network layers. Use the packet intercepted by Wireshark to explain the purpose of each layer in the 5-layer model. (10%)



5 Process & Applications　Hypertext Transfer Protocol
瀏覽器跟web server的溝通，包括GET/POST, URI, Host, 還有客戶端的一些meta
4 Transport　Transmission Control Protocol
保證資料可靠性及傳輸順序等問題，記錄來源和目標的port，時間戳記等等
3 Internet　Internet Protocol Version 4
將資料從來源ip傳輸到目標ip，為封包選擇路由
2 Link　Ethernet II…. (ARP)
兩個連接著的網卡(00:0c:29:1f:a1:05) => (00:50:56:f4:f9:03)
1 Physical　411 bytes on wire… on interface 0
物理上有3288 bits穿過網路線 (雖然這邊是VM模擬的)

2. In some scenario, the only protocol you can use in Transport Layer is UDP (maybe banned by admin). Now, you are asked to send an important file that cannot tolerate any data loss or re-ordering, is it possible to compensate for UDP's unreliability? How? (5%) (Extremely simple answer suffices)
既然UDP不可靠，就只好讓更上層來保證可靠性了，像HTTP/3使用UDP來傳輸，就是使用比UDP更上層的QUIC來糾錯
Reference: QUIC wiki
"each QUIC request is separately multiplexed and error corrected at the level of the QUIC driver"

System Administration

DateTime Format checker (10%)

```
#!/usr/bin/env bash
#   Version
#   Author: WildfootW
#   GitHub: github.com/WildfootW
#   Copyright (C) 2019 WildfootW All rights reserved.
#

#valid_date_split_char=("-" "/" ".")
valid_date_split_char=("-" "/" "\.") # mind grep regular expression
valid_time_split_char=(":")
valid_date_digit_number_0=(4 2 2)
valid_time_digit_number_0=(2 2 2)
valid_time_digit_number_1=(2 2)
```

```
if [ ! $# -eq 2 ]; then
    echo "Usage: ./format_check.sh [DATE] [TIME]"
    exit -1
fi

function check_format_sub()
{
    local -n origin_str=$1
    local -n valid_split_char=$2
    local -n valid_digit_number=$3
    #echo "$origin_str, ${valid_split_char[@]}, ${valid_digit_number[@]}"
    for symbol in ${valid_split_char[@]}; do
    #format_str="[0-9]{${valid_digit_number[0]}}"
    format_str="[0-9]\{${valid_digit_number[0]}\}"
        for index in $(seq 1 $((${#valid_digit_number[@]}-1))); do
            format_str+="$symbol"
            #format_str+="[0-9]{${valid_digit_number[$index]}}"
            format_str+="[0-9]\{${valid_digit_number[$index]}\}" # mind
grep regular expression
        done
        #echo $format_str
        if [[ $origin_str = `echo $origin_str | grep -o $format_str` ]];
then
            #echo $origin_str | grep -o $format_str
            #echo "format $format_str success"
            return 1
        fi
    done
    return 0
}

time_origin=$2
date_origin=$1

check_format_sub date_origin valid_date_split_char
valid_date_digit_number_0
date_ret_0=$?
check_format_sub time_origin valid_time_split_char
valid_time_digit_number_0
time_ret_0=$?
check_format_sub time_origin valid_time_split_char
```

```
valid_time_digit_number_1
time_ret_1=$?

if [ $date_ret_0 -eq 1 ]; then
    if [ $time_ret_0 -eq 1 ] || [ $time_ret_1 -eq 1 ]; then
        echo "$date_origin $time_origin"
        exit 0
    fi
fi
echo "Invalid"
exit 0
```

How close are you? (15%)

```
#!/usr/bin/env bash
#   Version
#   Author: WildfootW
#   GitHub: github.com/WildfootW
#   Copyright (C) 2019 WildfootW All rights reserved.
#

if [ ! $# -eq 1 ]; then
    echo "Usage ./rtt_test.sh [FILE]"
    exit -1
fi

get_ave_round_trip_time()
{
    local -n local_domain_name=$1
    local -n local_ave_time=$2
    local_ave_time=`ping -c 3 -q $local_domain_name 2> /dev/null | awk -v
FS="mdev = " '{ print $2 }' | awk -v FS="/" 'NF { print $1 }'` # NF for
ignore empty lines
}

FILEPATH="$1"
#cat "$FILEPATH"

declare -a result_list
while read line_origin; do
    line_header=`echo "$line_origin" | awk '{ print $1 }'`
    #echo "$line_header"
```

```
    if [[ "$line_header" = "" ]]; then
        continue
    elif [ "$line_header" = "##" ]; then
        continue
    elif [ "$line_header" = "#Server" ]; then
        server_domain=`echo "$line_origin" | awk '{ print $3 }'`
        ave_time=""
        get_ave_round_trip_time server_domain ave_time
        if [[ "$ave_time" != "" ]]; then
            result_list+=("$server_domain $ave_time")
        fi
    fi
done < $FILEPATH

# strange printf behavior
printf "%s\n" "${result_list[@]}" | sort -g -k 2
```

CSIE Analytics (25%)

```
#!/usr/bin/env bash
#   Version
#   Author: WildfootW
#   GitHub: github.com/WildfootW
#   Copyright (C) 2019 WildfootW All rights reserved.
#

# option parser
OPTION_N=10
while [[ ! $# -eq 0 ]]; do
    key="$1"
    shift
    case $key in
        -n)
            if [[ ! $1 =~ ^-?[0-9]+$ ]] && [[ ! $1 =~ ^-?[0-9]+\.?[0-9]+$
]]; then # if not a [10, -10, 10.5, -5.2]
                echo "Error: option requires an argument."
                exit 1
            elif [[ ! $1 =~ ^[0-9]+$ ]]; then # if not a positive integer
                echo "Error: line number must be positive integer."
                exit 1
```

```
            fi
            OPTION_N="$1"
            shift
            ;;
        *)
            if [[ $FILEPATH != "" ]] || [[ $key =~ ^-.*$ ]]; then
                echo "Usage: csie_analytics.sh [-n count] [filename]"
                exit 1
            fi
            FILEPATH="$key"
            if [[ ! -e $FILEPATH ]]; then
                echo "Error: log file does not exist."
                exit 1
            fi
            ;;
    esac
done
if [[ $FILEPATH == "" ]]; then
    echo "Usage: csie_analytics.sh [-n count] [filename]"
    exit 1
fi
#echo "OPTION_N = ${OPTION_N}"
#echo "FILEPATH = ${FILEPATH}"

result=`cat "$FILEPATH" | awk -v FS="(GET|POST) " '{ print $2 }' | awk -v
FS="(\?| HTTP\/[0-9])" '{ print $1 }' | sort | uniq -c | sort -g -r -k 1`

declare -a path_list
declare -a query_times_list
total_query_times=0
while read -r line; do
    query_times=`echo $line | awk '{ print $1 }'`
    path_list+=(`echo $line | awk '{ print $2 }'`)
    query_times_list+=($query_times)
    total_query_times=$(($total_query_times+$query_times))
done <<< "$result"

printf "%-35s %-10s %s\n" "Path" "Times" "Percentage"
for((i = 0;i < ${#path_list[@]} && i < $OPTION_N;++i)); do
    percentage=`bc -l <<< "${query_times_list[$i]}*100/$total_query_times"`
    printf "%-35s %-10s %-2.2f%%\n" ${path_list[$i]}
${query_times_list[$i]} $percentage
```

```
done
```