

NASA HW1

b09902004 郭懷元

Network Administration

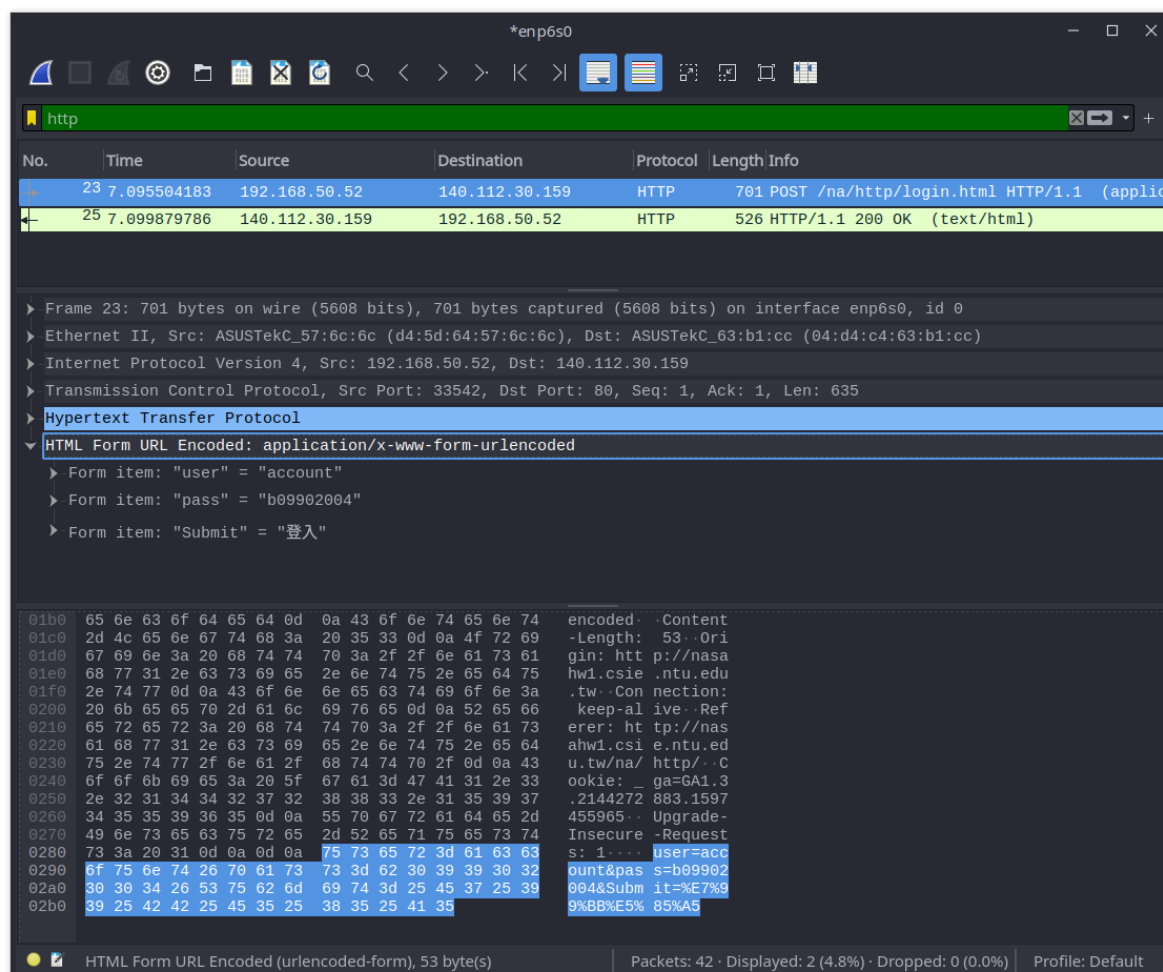
野生的密碼難道會在網路上赤裸地奔馳著？

1.

Reference:

http://www.cs.nccu.edu.tw/~jang/teaching/CompNet_files/Wireshark-%E5%9F%BA%E7%A4%8E%E6%95%99%E5%AD%B8.pdf

Type `http` in display filter to make it easier to find the packet.



2.

Reference:

<https://zh.wikipedia.org/zh-tw/%E8%B6%85%E6%96%87%E6%9C%AC%E4%BC%A0%E8%B%E%93%E5%AE%89%E5%85%A8%E5%8D%8F%E8%AE%AE>

The packet can't be found. Because we are accessing 登入界面 with https, the content of packets sent between the server and pc is encrypted. Therefore it is impossible to identify which packet contains my account and password if we only look at the content of each packet.

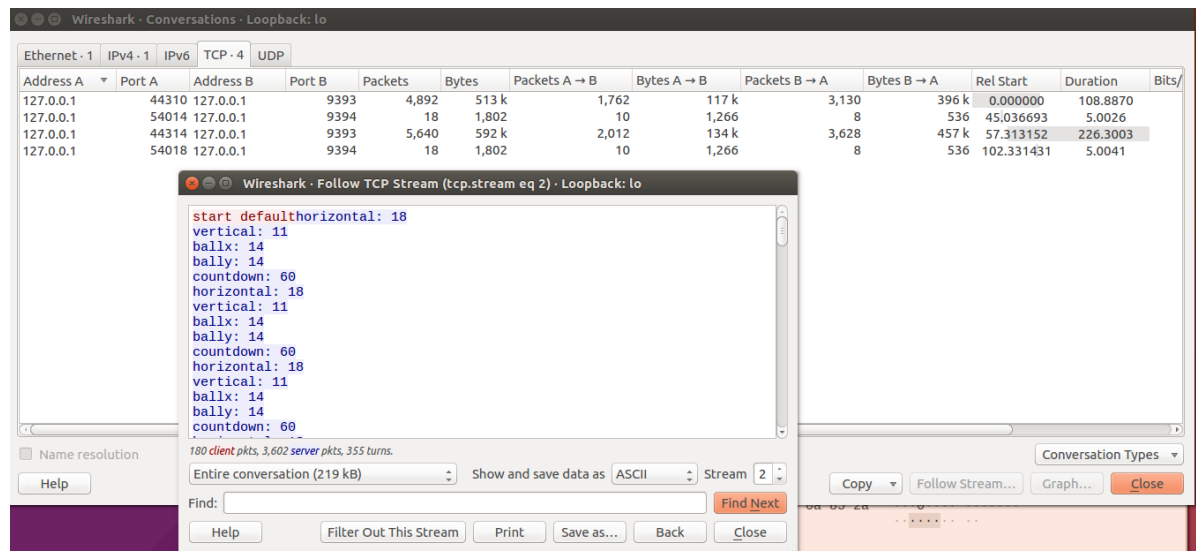
好玩遊戲也有暗潮洶湧的一面

1.

Reference:

b09902011 陳可邦

In Wireshark, go to `statistics -> conversations`, and we can see something like the image below after playing 2 games while Wireshark is running. Select the first row and click `Follow Stream...`.



After reading the conversation between client and server, we know the game works this way:

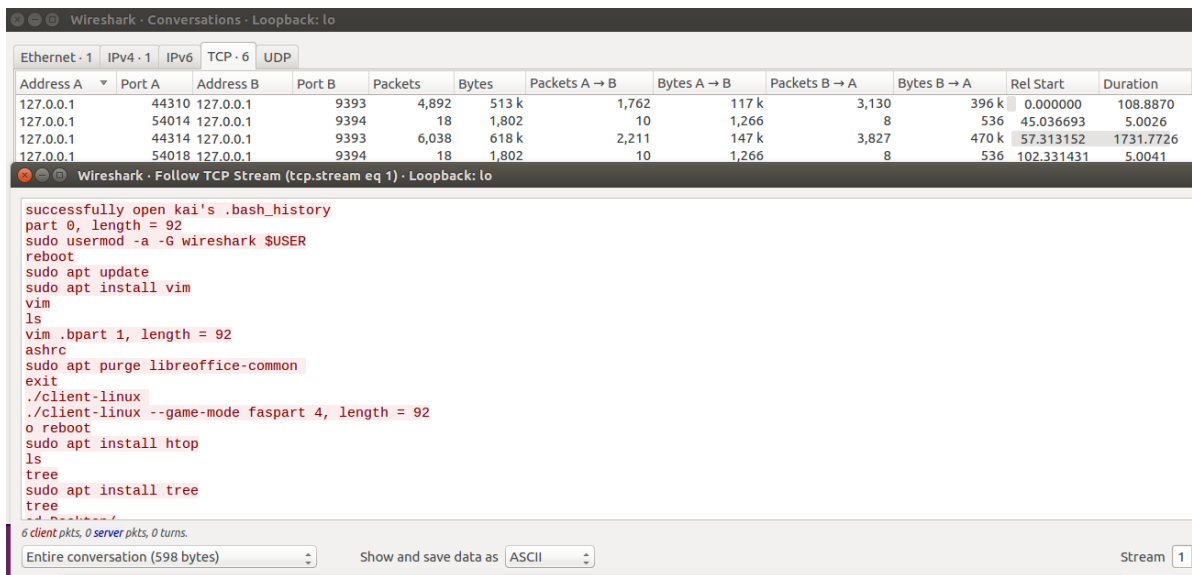
0. The client and the server establish a TCP connection. All conversation are made on this TCP connection.
1. The client sends `start <game mode>` to the server at `localhost:9393`. The default game mode is `default`, other available game modes are `fast` and `double`.
2. The server will send horizontal blocks' position, vertical blocks' position, ball's cordinate, and a countdown. The message is sent at a fixed frequency.
3. While the server updates information to let client render, the client will send our moves to host, so that the server know to update blocks' position.
4. If the countdown goes to 0, the server will send `win`. If the ball hits the wall, the server will send `lose`. When client receives game result, the result will be printed and game will stop running.
5. The TCP connection is terminated only when the process is killed.

2.

Reference:

b09902011 陳可邦

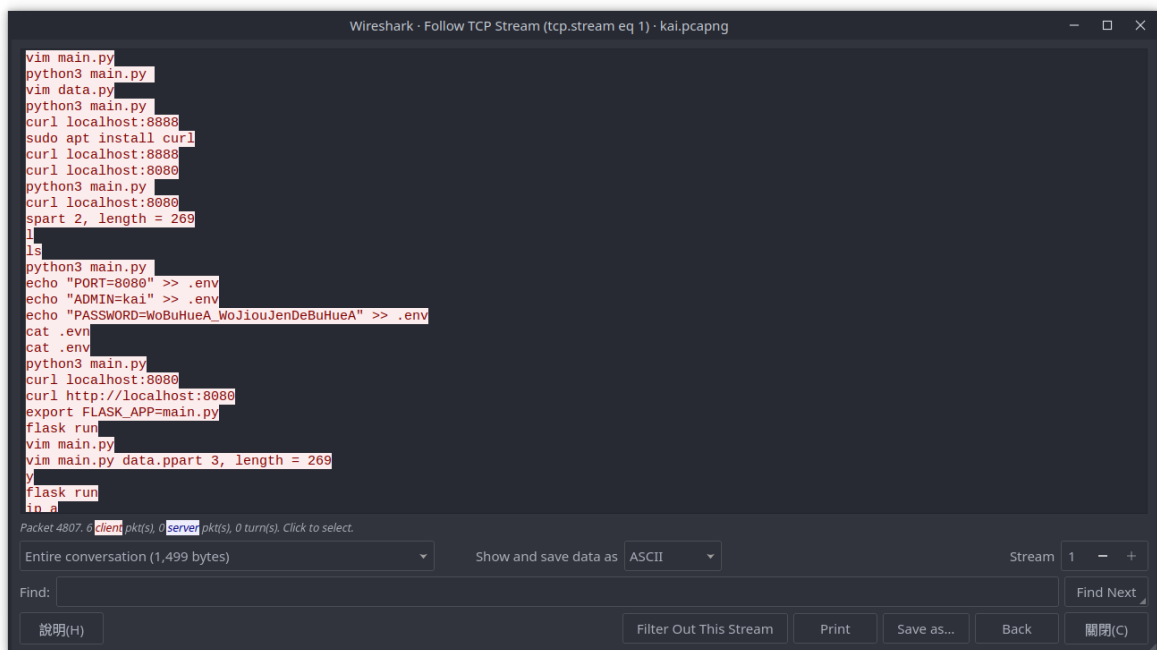
In the `conversation` window, choose the conversation to `127.0.0.1:9394`



The server will send a message that includes the line `give me secret\n` at a random time during game at port `localhost:9393`. Then the client will send content of `~/.bash_history` to the server at `localhost:9394`. `.bash_history` contents every line you entered in bash before, and it might contains some sensitive information such as passwords and api keys.

3.

Go to `statistics` -> `conversations`, find the conversation to `127.0.0.1:9394`.



We can see that the password is `WoBuHueA_WoJiouJenDeBuHueA`.

4.

Reference:

b09902011 陳可邦

<https://clay-atlas.com/blog/2019/10/15/python-chinese-tutorial-socket-tcp-ip/>

Flag: `HW1{d0_y0u_knovv_wH0_KaienLin_1s?}`

```
vm [執行中] - Oracle VM VirtualBox
kai@ubuntu:~$ python client-fast.py
win HW1{d0_y0u_knovv_wH0_KaienLin_1s?}
kai@ubuntu:~$
```

Since the client only handles transferring user inputs to the server, we can still play the game without `client-linux`. I choose python to communicate with the server and play games, because I am familiar with it and Ubuntu already has it. We can know what to send to the server by simply looking at past conversation when playing game with `client-linux`. The script I use is [here](#)

5.

Reference:

b09902011 陳可邦

<https://clay-atlas.com/blog/2019/10/15/python-chinese-tutorial-socket-tcp-ip/>

Flag: `HW1{Dou8l3_b@ll_d0uB1e_Fun!}`

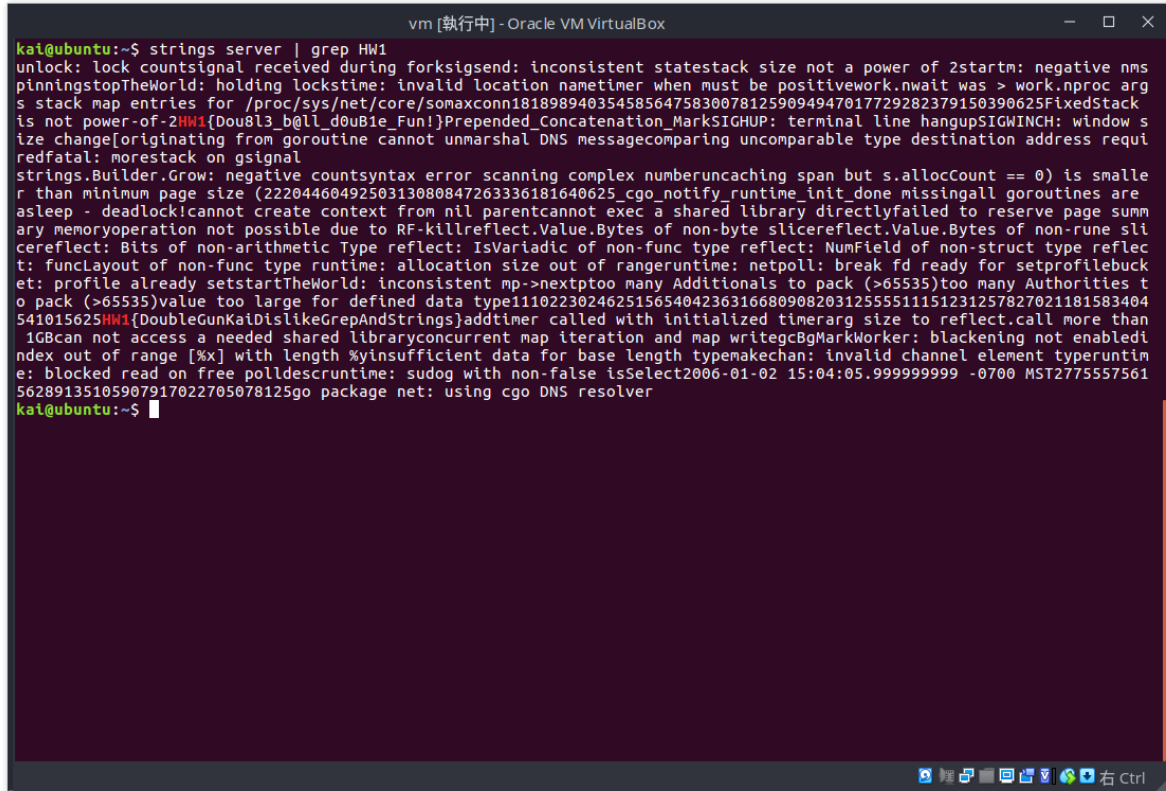
```
vm [執行中] - Oracle VM VirtualBox
kai@ubuntu:~$ python client-double.py
win HW1{Dou8l3_b@ll_d0uB1e_Fun!}
kai@ubuntu:~$
```

The approach is basically the same as last one. The only thing I changed is game-playing strategy because there are two balls. The script is [here](#)

An alternative approach is by looking at the server's binaries.

```
netstat -tulpn
htop
sudo cp /root/server ~
strings server | grep HW1
```

Using `netstat -tulpn` & `htop`, we can find out that the server is at `/root/server`. Then `sudo cp /root/server ~` and `strings server | grep HW1`. The flag for double mode and the fake flag is here.



```
vm [執行中] - Oracle VM VirtualBox
kai@ubuntu:~$ strings server | grep HW1
unlock: lock countsignal received during forksigsend: inconsistent statestack size not a power of 2startm: negative nms
pinningstopTheWorld: holding locktime: invalid location nametimer when must be positivework.nwait was > work.nproc arg
s stack map entries for /proc/sys/net/core/somaxconn18189894035458564758300781259094947017729282379150390625FixedStack
is not power-of-2HW1{Dou8l3_b@ll_d0u81e_Fun!}Prepended_Concatenation_MarksIGHUP: terminal line hangupSIGWINCH: window s
ize change[originating from goroutine cannot unmarshal DNS messagecomparing uncomparable type destination address requi
redfatal: morestack on gsignal
strings.Builder.Grow: negative countsyntax error scanning complex numberuncaching span but s.allocCount == 0) is smalle
r than minimum page size (2220446049250313080847263336181640625_cgo_notify_runtime_init_done missingall goroutines are
asleep - deadlock!cannot create context from nil parentcannot exec a shared library directlyfailed to reserve page summ
ary memoryoperation not possible due to RF-killreflect.Value.Bytes of non-byte slicereflect.Value.Bytes of non-rune sli
cereflect: Bits of non-arithmetic Type reflect: IsVariadic of non-func type reflect: NumField of non-struct type reflec
t: funcLayout of non-func type runtime: allocation size out of rangerruntime: netpoll: break fd ready for setprofilebuck
et: profile already setstartTheWorld: inconsistent mp->nextptoo many Additionals to pack (>65535)too many Authorities t
o pack (>65535)value too large for defined data type11022302462515654042363166809082031255551151231257827021181583404
541015625HW1{DoubleGunKaiDislikeGrepAndStrings}addtimer called with initialized timerarg size to reflect.call more than
1GBcan not access a needed shared libraryconcurrent map iteration and map writegcBgMarkWorker: blackening not enabledi
ndex out of range [%x] with length %yinsufficient data for base length typemakechan: invalid channel element typeruntim
e: blocked read on free polldescruntime: sudog with non-false isSelect2006-01-02 15:04:05.999999999 -0700 MST2775557561
56289135105907917022705078125go package net: using cgo DNS resolver
kai@ubuntu:~$
```

這麼多的網路協定要是能全部都認識的話該有多好

1.

Reference:

http://linux.vbird.org/linux_server/0110network_basic.php#tcpip_network_icmp

icmp-packet.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
196	33.061718891	192.168.50.52	172.217.160.110	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in
198	33.062093054	192.168.50.52	192.168.50.1	ICMP	137	Destination unreachable (Port unreachable)
199	33.065992904	172.217.160.110	192.168.50.52	ICMP	98	Echo (ping) reply id=0x0001, seq=1/256, ttl=116 (request
204	33.069839153	192.168.50.52	192.168.50.1	ICMP	166	Destination unreachable (Port unreachable)
206	34.063708827	192.168.50.52	172.217.160.110	ICMP	98	Echo (ping) request id=0x0001, seq=2/512, ttl=64 (reply in
207	34.070047815	172.217.160.110	192.168.50.52	ICMP	98	Echo (ping) reply id=0x0001, seq=2/512, ttl=116 (request
210	35.065113384	192.168.50.52	172.217.160.110	ICMP	98	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in
211	35.070087710	172.217.160.110	192.168.50.52	ICMP	98	Echo (ping) reply id=0x0001, seq=3/768, ttl=116 (request
213	36.066151633	192.168.50.52	172.217.160.110	ICMP	98	Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (reply in

▶ Frame 206: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp6s0, id 0
 ▶ Ethernet II, Src: ASUSTekC_57:6c:6c (d4:5d:64:57:6c:6c), Dst: ASUSTekC_63:b1:cc (04:d4:c4:63:b1:cc)
 ▶ Internet Protocol Version 4, Src: 192.168.50.52, Dst: 172.217.160.110
 ▶ Internet Control Message Protocol

```

0000  04 d4 c4 63 b1 cc d4 5d 64 57 6c 6c 08 00 45 00  ...c... ] dwll...E
0010  00 54 70 cb 40 00 40 01 89 b9 c0 a8 32 34 ac d9  .Tp @ @  ....24..
0020  a0 6e 08 00 e6 11 00 01 00 02 4f 50 40 60 00 00  .n.....-OP@...
0030  00 00 ba 67 09 00 00 00 00 00 10 11 12 13 14 15  .g.....
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25  ..... !"#$$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35  &'()*+,-./012345
0060  36 37                                     67
  
```

Internet Control Message Protocol: Protocol | Packets: 355 · Displayed: 15 (4.2%) · Dropped: 0 (0.0%) | Profile: Default

icmp-packet.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
196	33.061718891	192.168.50.52	172.217.160.110	ICMP	98	Echo (ping) request id=0x0001, seq=1/256, ttl=64 (reply in
198	33.062093054	192.168.50.52	192.168.50.1	ICMP	137	Destination unreachable (Port unreachable)
199	33.065992904	172.217.160.110	192.168.50.52	ICMP	98	Echo (ping) reply id=0x0001, seq=1/256, ttl=116 (request
204	33.069839153	192.168.50.52	192.168.50.1	ICMP	166	Destination unreachable (Port unreachable)
206	34.063708827	192.168.50.52	172.217.160.110	ICMP	98	Echo (ping) request id=0x0001, seq=2/512, ttl=64 (reply in
207	34.070047815	172.217.160.110	192.168.50.52	ICMP	98	Echo (ping) reply id=0x0001, seq=2/512, ttl=116 (request
210	35.065113384	192.168.50.52	172.217.160.110	ICMP	98	Echo (ping) request id=0x0001, seq=3/768, ttl=64 (reply in
211	35.070087710	172.217.160.110	192.168.50.52	ICMP	98	Echo (ping) reply id=0x0001, seq=3/768, ttl=116 (request
213	36.066151633	192.168.50.52	172.217.160.110	ICMP	98	Echo (ping) request id=0x0001, seq=4/1024, ttl=64 (reply in

▶ Frame 207: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface enp6s0, id 0
 ▶ Ethernet II, Src: ASUSTekC_63:b1:cc (04:d4:c4:63:b1:cc), Dst: ASUSTekC_57:6c:6c (d4:5d:64:57:6c:6c)
 ▶ Internet Protocol Version 4, Src: 172.217.160.110, Dst: 192.168.50.52
 ▶ Internet Control Message Protocol

```

0000  d4 5d 64 57 6c 6c 04 d4 c4 63 b1 cc 08 00 45 00  .]dwll...c...E
0010  00 54 00 00 00 00 74 01 06 85 ac d9 a0 6e c0 a8  .T...t.....n..
0020  32 34 00 00 ee 11 00 01 00 02 4f 50 40 60 00 00  24.....-OP@...
0030  00 00 ba 67 09 00 00 00 00 00 10 11 12 13 14 15  .g.....
0040  16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25  ..... !"#$$%
0050  26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35  &'()*+,-./012345
0060  36 37                                     67
  
```

Internet Control Message Protocol: Protocol | Packets: 355 · Displayed: 15 (4.2%) · Dropped: 0 (0.0%) | Profile: Default

ICMP is a protocol working in network layer. The main task of ICMP is sending error messages and information about connection status. Both `ping` and `traceroute` rely on ICMP to work.

2.

Reference:

https://en.wikipedia.org/wiki/Domain_Name_System

dns-packet.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
2221	-6.129748207	192.168.50.52	192.168.50.1	DNS	81	Standard query 0x30bc A github.com OPT
2222	-6.129709314	192.168.50.52	192.168.50.1	DNS	81	Standard query 0xd539 A github.com OPT
2223	-6.129646386	192.168.50.52	192.168.50.1	DNS	81	Standard query 0xae9e AAAA github.com OPT
2224	-6.129629604	192.168.50.52	192.168.50.1	DNS	81	Standard query 0x5cc0 AAAA github.com OPT
2225	-6.124210308	192.168.50.1	192.168.50.52	DNS	97	Standard query response 0x30bc A github.com A 13.114.40.48 0
2226	-6.124118115	192.168.50.1	192.168.50.52	DNS	97	Standard query response 0xd539 A github.com A 52.69.186.44 0
2227	-6.124104670	192.168.50.52	192.168.50.1	ICMP	125	Destination unreachable (Port unreachable)
2228	-6.124019840	192.168.50.1	192.168.50.52	DNS	165	Standard query response 0xae9e AAAA github.com SOA ns-1707.a
2229	-6.123870469	192.168.50.1	192.168.50.52	DNS	146	Standard query response 0x5cc0 AAAA github.com SOA dns1.p08.m

▶ Frame 2221: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface enp6s0, id 0
 ▶ Ethernet II, Src: ASUSTekC_57:6c:6c (d4:5d:64:57:6c:6c), Dst: ASUSTekC_63:b1:cc (04:d4:c4:63:b1:cc)
 ▶ Internet Protocol Version 4, Src: 192.168.50.52, Dst: 192.168.50.1
 ▶ User Datagram Protocol, Src Port: 51080, Dst Port: 53
 ▶ Domain Name System (query)

0000 04 d4 c4 63 b1 cc d4 5d 64 57 6c 6c 08 00 45 00 ...c...] dwll...E
 0010 00 43 3d 8d 40 00 40 11 17 97 c0 a8 32 34 c0 a8 ...C=@@...24...
 0020 32 01 c7 88 00 35 00 2f e5 c6 30 bc 01 00 00 01 2...5 / ...0...
 0030 00 00 00 00 00 01 06 67 69 74 68 75 62 03 63 6fg ithub.co
 0040 6d 00 00 01 00 01 00 00 29 02 00 00 00 00 00 00 m.....).....
 0050 00

Domain Name System: Protocol | Packets: 2878 · Displayed: 105 (3.6%) · Dropped: 0 (0.0%) | Profile: Default

dns-packet.pcapng

No.	Time	Source	Destination	Protocol	Length	Info
2221	-6.129748207	192.168.50.52	192.168.50.1	DNS	81	Standard query 0x30bc A github.com OPT
2222	-6.129709314	192.168.50.52	192.168.50.1	DNS	81	Standard query 0xd539 A github.com OPT
2223	-6.129646386	192.168.50.52	192.168.50.1	DNS	81	Standard query 0xae9e AAAA github.com OPT
2224	-6.129629604	192.168.50.52	192.168.50.1	DNS	81	Standard query 0x5cc0 AAAA github.com OPT
2225	-6.124210308	192.168.50.1	192.168.50.52	DNS	97	Standard query response 0x30bc A github.com A 13.114.40.48 0
2226	-6.124118115	192.168.50.1	192.168.50.52	DNS	97	Standard query response 0xd539 A github.com A 52.69.186.44 0
2227	-6.124104670	192.168.50.52	192.168.50.1	ICMP	125	Destination unreachable (Port unreachable)
2228	-6.124019840	192.168.50.1	192.168.50.52	DNS	165	Standard query response 0xae9e AAAA github.com SOA ns-1707.a
2229	-6.123870469	192.168.50.1	192.168.50.52	DNS	146	Standard query response 0x5cc0 AAAA github.com SOA dns1.p08.m

▶ Frame 2225: 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface enp6s0, id 0
 ▶ Ethernet II, Src: ASUSTekC_63:b1:cc (04:d4:c4:63:b1:cc), Dst: ASUSTekC_57:6c:6c (d4:5d:64:57:6c:6c)
 ▶ Internet Protocol Version 4, Src: 192.168.50.1, Dst: 192.168.50.52
 ▶ User Datagram Protocol, Src Port: 53, Dst Port: 51080
 ▶ Domain Name System (response)

0000 d4 5d 64 57 6c 6c 04 d4 c4 63 b1 cc 08 00 45 00 ...]dwll...c...E
 0010 00 53 09 e2 40 00 40 11 4b 32 c0 a8 32 01 c0 a8 ...S.@@..K2..2...
 0020 32 34 00 35 c7 88 00 3f a6 41 30 bc 81 80 00 01 24 5...?..A0...
 0030 00 01 00 00 00 01 06 67 69 74 68 75 62 03 63 6fg ithub.co
 0040 6d 00 00 01 00 01 c0 0c 00 01 00 01 00 00 2b m.....+
 0050 00 04 0d 72 28 30 00 00 29 04 d0 00 00 00 00 00 ...r(0...).....
 0060 00

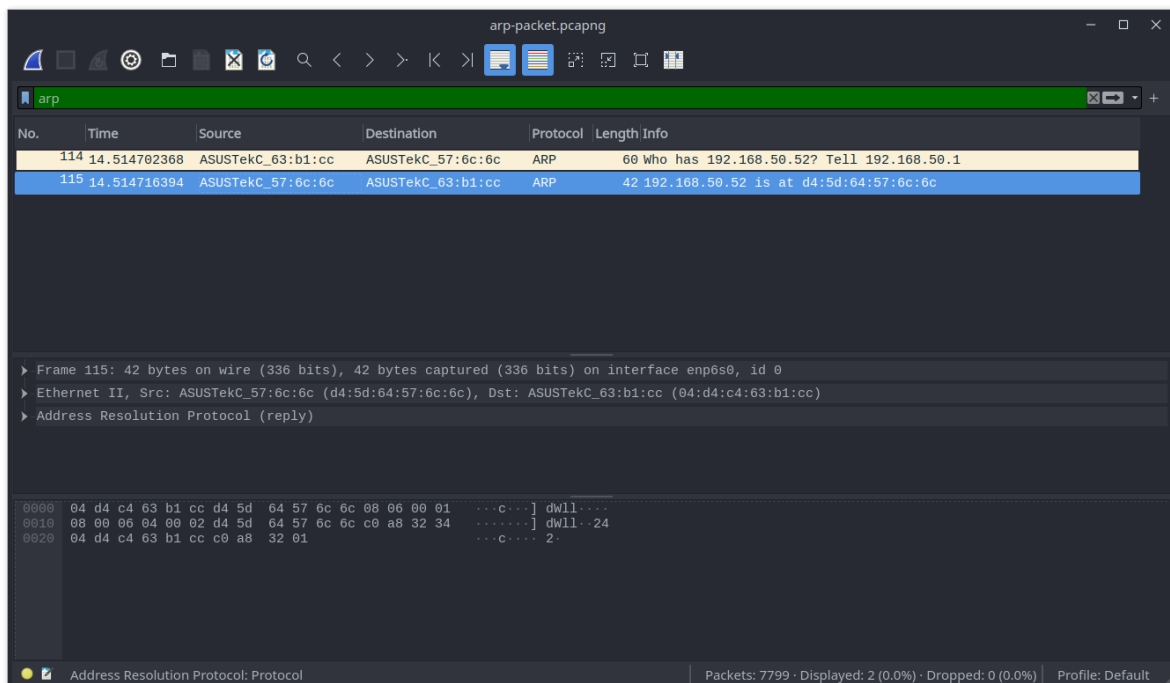
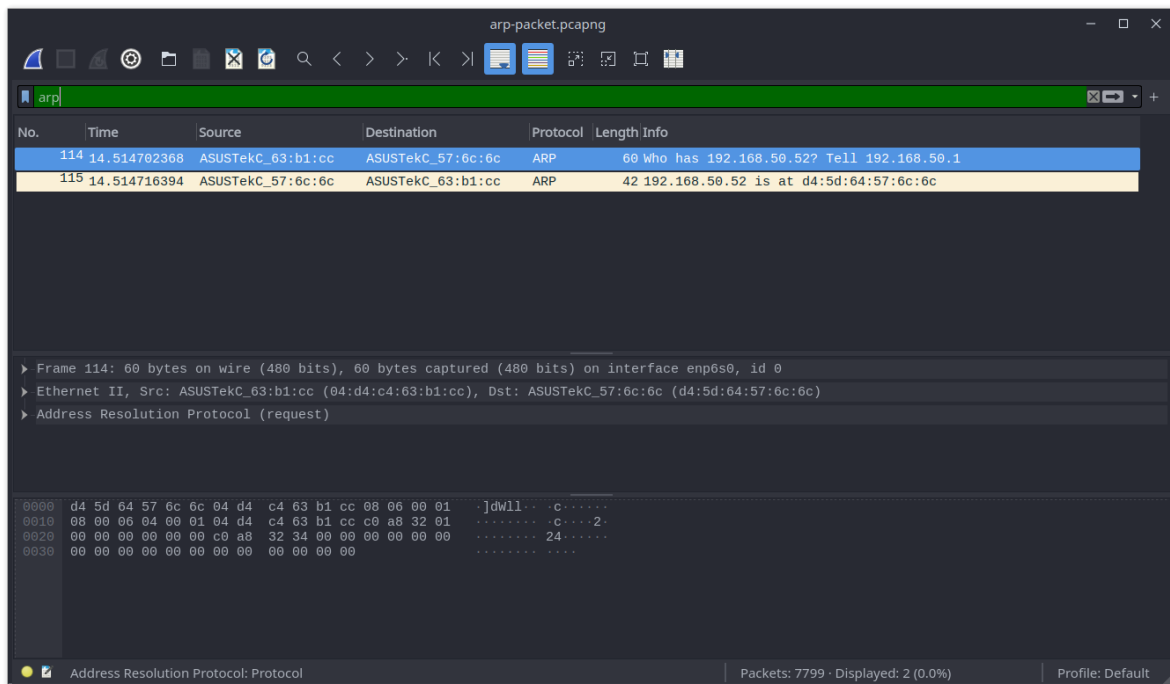
Domain Name System: Protocol | Packets: 2878 · Displayed: 105 (3.6%) · Dropped: 0 (0.0%) | Profile: Default

DNS works on application layer. DNS is used to translate human-readable hostnames (e.g. `www.ntu.edu.tw`) to IP addresses (e.g. `140.112.8.116`) that network devices use. It's an essential part of the Internet we use today.

3.

Reference:

<https://zh.wikipedia.org/wiki/%E5%9C%B0%E5%9D%80%E8%A7%A3%E6%9E%90%E5%8D%8F%E8%AE%AE>



ARP is a protocol working on link layer. In TCP/IP protocols, network layer and transport layer uses IP addresses, but Ethernet requires MAC addresses to transfer data. ARP handles the transition between IP addresses and MAC addresses to make everything work.

4.

Reference:

http://linux.vbird.org/linux_server/0340dhcp.php

dhcpcap.pcapng

dhcpcap

No.	Time	Source	Destination	Protocol	Length	Info
254	57.333887550	0.0.0.0	255.255.255.255	DHCP	345	DHCP Request - Transaction ID 0xba3b9a9f
261	59.333745408	0.0.0.0	255.255.255.255	DHCP	345	DHCP Discover - Transaction ID 0x39f6fcea
262	59.334731321	192.168.50.1	192.168.50.52	DHCP	342	DHCP Offer - Transaction ID 0x39f6fcea
263	59.334759985	0.0.0.0	255.255.255.255	DHCP	351	DHCP Request - Transaction ID 0x39f6fcea
264	59.335837119	192.168.50.1	192.168.50.52	DHCP	358	DHCP ACK - Transaction ID 0x39f6fcea

Frame 261: 345 bytes on wire (2760 bits), 345 bytes captured (2760 bits) on interface enp6s0, id 0

Ethernet II, Src: ASUSTeK 57:6c:6c (d4:5d:64:57:6c:6c), Dst: Broadcast (ff:ff:ff:ff:ff:ff)

Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255

User Datagram Protocol, Src Port: 68, Dst Port: 67

Dynamic Host Configuration Protocol (Discover)

0000 ff ff ff ff ff ff d4 5d 64 57 6c 6c 08 00 45 c0] dwll...E
0010 01 4b 00 00 40 00 40 11 38 e3 00 00 00 00 ff ff -K-@-@- 8-...
0020 ff ff 00 44 00 43 01 37 25 ee 01 01 06 00 39 f6 --D-C-7 %-...9-
0030 fc ea 00 01 00 00 00 00 00 00 00 00 00 00 00 00
0040 00 00 00 00 00 00 d4 5d 64 57 6c 6c 00 00 00 00] dwll...
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Dynamic Host Configuration Protocol: Protocol

Packets: 1003 · Displayed: 5 (0.5%) · Dropped: 0 (0.0%) · Profile: Default

dhcpcap.pcapng

dhcpcap

No.	Time	Source	Destination	Protocol	Length	Info
254	57.333887550	0.0.0.0	255.255.255.255	DHCP	345	DHCP Request - Transaction ID 0xba3b9a9f
261	59.333745408	0.0.0.0	255.255.255.255	DHCP	345	DHCP Discover - Transaction ID 0x39f6fcea
262	59.334731321	192.168.50.1	192.168.50.52	DHCP	342	DHCP Offer - Transaction ID 0x39f6fcea
263	59.334759985	0.0.0.0	255.255.255.255	DHCP	351	DHCP Request - Transaction ID 0x39f6fcea
264	59.335837119	192.168.50.1	192.168.50.52	DHCP	358	DHCP ACK - Transaction ID 0x39f6fcea

Frame 262: 342 bytes on wire (2736 bits), 342 bytes captured (2736 bits) on interface enp6s0, id 0

Ethernet II, Src: ASUSTeK 63:b1:cc (04:d4:c4:63:b1:cc), Dst: ASUSTeK 57:6c:6c (d4:5d:64:57:6c:6c)

Internet Protocol Version 4, Src: 192.168.50.1, Dst: 192.168.50.52

User Datagram Protocol, Src Port: 67, Dst Port: 68

Dynamic Host Configuration Protocol (Offer)

0000 d4 5d 64 57 6c 6c 04 d4 c4 63 b1 cc 08 00 45 00 .]dwll...-c-...E
0010 01 48 66 03 00 00 40 11 2e 1c c0 a8 32 01 c0 a8 -HF-@-@-...2-..
0020 32 34 00 43 00 44 01 34 bc 56 02 01 06 00 39 f6 24-C-D-4-V-...9-
0030 fc ea 00 01 00 00 00 00 00 00 c0 a8 32 34 c0 a8-24-..
0040 32 01 00 00 00 00 d4 5d 64 57 6c 6c 00 00 00 00 2-.....] dwll...
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Dynamic Host Configuration Protocol: Protocol

Packets: 1003 · Displayed: 5 (0.5%) · Dropped: 0 (0.0%) · Profile: Default

The screenshot shows a Wireshark packet capture of a DHCP Request. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
254	57.333887550	0.0.0.0	255.255.255.255	DHCP	345	DHCP Request - Transaction ID 0xba3b9a9f
261	59.333745408	0.0.0.0	255.255.255.255	DHCP	345	DHCP Discover - Transaction ID 0x39f6fcea
262	59.334731321	192.168.50.1	192.168.50.52	DHCP	342	DHCP Offer - Transaction ID 0x39f6fcea
263	59.334759985	0.0.0.0	255.255.255.255	DHCP	351	DHCP Request - Transaction ID 0x39f6fcea
264	59.335837119	192.168.50.1	192.168.50.52	DHCP	358	DHCP ACK - Transaction ID 0x39f6fcea

The packet details for the selected DHCP Request (No. 263) are:

- Frame 263: 351 bytes on wire (2808 bits), 351 bytes captured (2808 bits) on interface enp6s0, id 0
- Ethernet II, Src: ASUSTekC_57:6c:6c (d4:5d:64:57:6c:6c), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Internet Protocol Version 4, Src: 0.0.0.0, Dst: 255.255.255.255
- User Datagram Protocol, Src Port: 68, Dst Port: 67
- Dynamic Host Configuration Protocol (Request)

The packet bytes section shows the raw data in hexadecimal and ASCII.

The screenshot shows a Wireshark packet capture of a DHCP ACK. The packet list table is as follows:

No.	Time	Source	Destination	Protocol	Length	Info
254	57.333887550	0.0.0.0	255.255.255.255	DHCP	345	DHCP Request - Transaction ID 0xba3b9a9f
261	59.333745408	0.0.0.0	255.255.255.255	DHCP	345	DHCP Discover - Transaction ID 0x39f6fcea
262	59.334731321	192.168.50.1	192.168.50.52	DHCP	342	DHCP Offer - Transaction ID 0x39f6fcea
263	59.334759985	0.0.0.0	255.255.255.255	DHCP	351	DHCP Request - Transaction ID 0x39f6fcea
264	59.335837119	192.168.50.1	192.168.50.52	DHCP	358	DHCP ACK - Transaction ID 0x39f6fcea

The packet details for the selected DHCP ACK (No. 264) are:

- Frame 264: 358 bytes on wire (2864 bits), 358 bytes captured (2864 bits) on interface enp6s0, id 0
- Ethernet II, Src: ASUSTekC_63:b1:cc (04:d4:c4:63:b1:cc), Dst: ASUSTekC_57:6c:6c (d4:5d:64:57:6c:6c)
- Internet Protocol Version 4, Src: 192.168.50.1, Dst: 192.168.50.52
- User Datagram Protocol, Src Port: 67, Dst Port: 68
- Dynamic Host Configuration Protocol (ACK)

The packet bytes section shows the raw data in hexadecimal and ASCII.

DHCP works on application layer. A DHCP server automatically offers parameters for network configurations (such as IP address) to devices within the same LAN. It makes adding new devices in to the current network much easier.

System Administration

Reference:

Lab - shell script's slides

http://linux.vbird.org/linux_basic/0340bashshell-scripts.php

<https://blog.techbridge.cc/2019/11/15/linux-shell-script-tutorial/>

<https://gary840227.medium.com/linux-bash-array-%E4%BB%8B%E7%B4%B9-6e30ffe87978>

<https://stackoverflow.com/questions/806906/how-do-i-test-if-a-variable-is-a-number-in-bash>

and dozens pages that I forgot to copy the url

Task 1 - Argument Parser and Checker

The first part is to parse the parameters, I use `case` to implement this part. Checking filename format can be done with some simple regular expression.

For the validity checks, I implement those with some built-in arguments like `-n`, `-z`, `-e`, `-f`, and `-r`.

Task 2 - Crawler and Filter

Parsing parameters here is much easier, since everything is ordered in a fixed way. I use `sed` to get rid of `$` signs at the end.

Translating relative paths to absolute paths is done with `realpath`, and checking the directory of a file is done with `dirname`.

I use `curl` with `-L` option to follow the redirection, and with `-s` to disable `curl`'s progress bar. Parsing the raw html file is done by using regular expressions and `sed`. Getting the filename without path and extension is done by using `basename` and `sed`.

For the sorting part, `LC_ALL=C` is to make sure strings are ordered by dictionary order.

Task 3 - Analyzer

The cases where `comp=0` or `comp=1 && target isn't empty` are rather simple, just use `cut` to extract filenames from input file and sort them. As for the case where `comp=1 && target is empty`, I use the provided algorithm and pseudo code to implement. The checkpoint can be easily done using `$right` and `$left` that we create when building forward star.