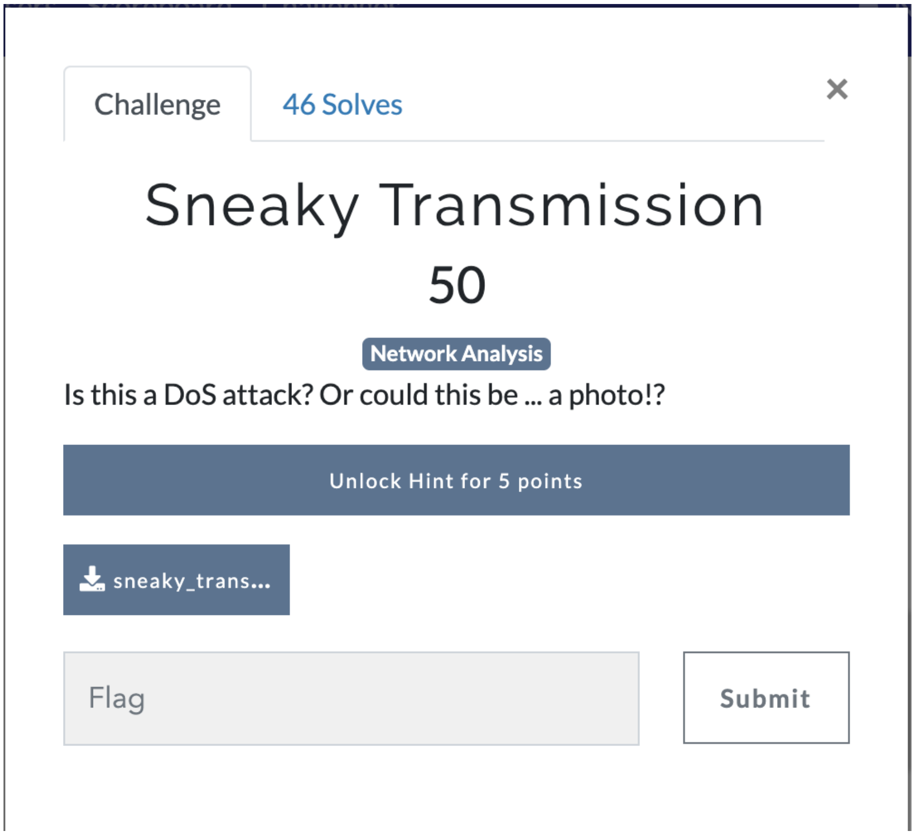
Networking Challenge

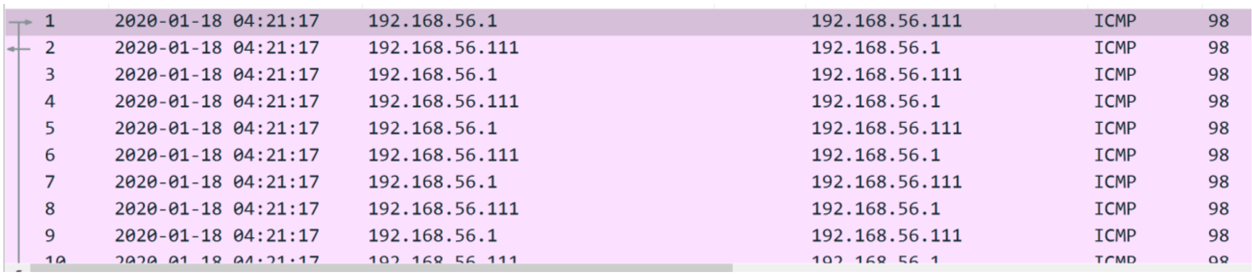


Sneaky Transmission

50

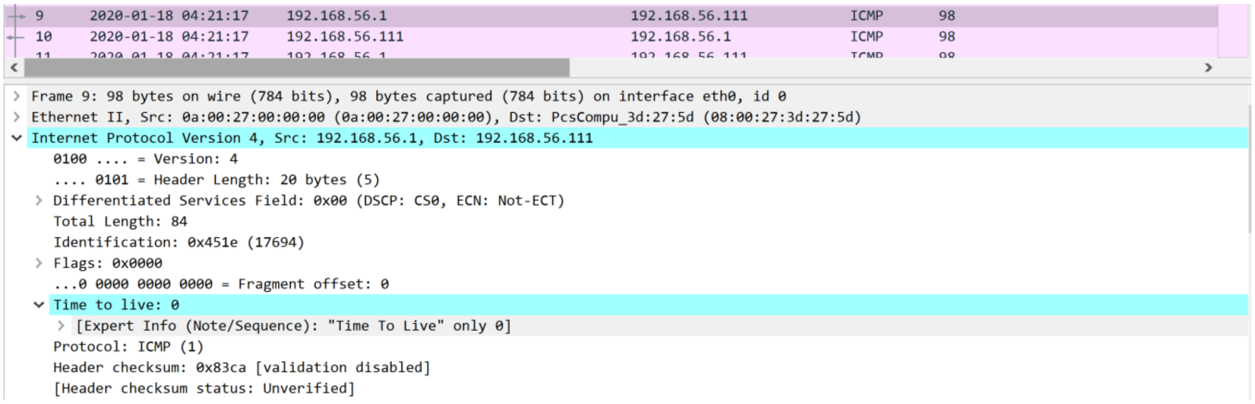
Is this a DoS attack? Or could this be ... a photo!?

Answer: We're given a PCAP lets open it up.



```
1    2020-01-18 04:21:17   192.168.56.1     192.168.56.111   ICMP   98
2    2020-01-18 04:21:17   192.168.56.111   192.168.56.1     ICMP   98
3    2020-01-18 04:21:17   192.168.56.1     192.168.56.111   ICMP   98
4    2020-01-18 04:21:17   192.168.56.111   192.168.56.1     ICMP   98
5    2020-01-18 04:21:17   192.168.56.1     192.168.56.111   ICMP   98
6    2020-01-18 04:21:17   192.168.56.111   192.168.56.1     ICMP   98
7    2020-01-18 04:21:17   192.168.56.1     192.168.56.111   ICMP   98
8    2020-01-18 04:21:17   192.168.56.111   192.168.56.1     ICMP   98
9    2020-01-18 04:21:17   192.168.56.1     192.168.56.111   ICMP   98
10   2020-01-18 04:21:17   192.168.56.111   192.168.56.1     ICMP   98
```

Looks like all ICMP traffic. Looking at frame 9 we see something strange.



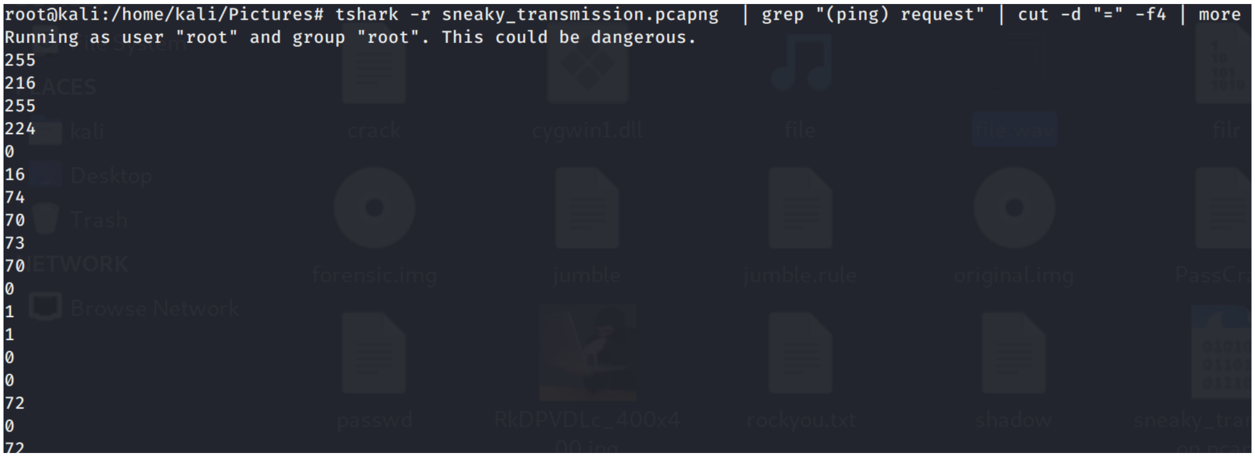
```
9    2020-01-18 04:21:17   192.168.56.1     192.168.56.111   ICMP   98
10   2020-01-18 04:21:17   192.168.56.111   192.168.56.1     ICMP   98
11   2020-01-18 04:21:17   192.168.56.1     192.168.56.111   ICMP   98

> Frame 9: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface eth0, id 0
> Ethernet II, Src: 0a:00:27:00:00:00 (0a:00:27:00:00:00), Dst: PcsCompu_3d:27:5d (08:00:27:3d:27:5d)
v Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.111
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0x451e (17694)
  > Flags: 0x0000
    ...0 0000 0000 0000 = Fragment offset: 0
  v Time to live: 0
    > [Expert Info (Note/Sequence): "Time To Live" only 0]
    Protocol: ICMP (1)
    Header checksum: 0x83ca [validation disabled]
    [Header checksum status: Unverified]
```

Why is the ttl 0. Looking at some of the other ICMP request it looks like the ttl is different for all. Let's extract these values and see what we get.

WE'll use tshark to complete this.

Command: tshark -r sneaky_transmission.pcapng | grep "(ping) request" | cut -d "=" -f4

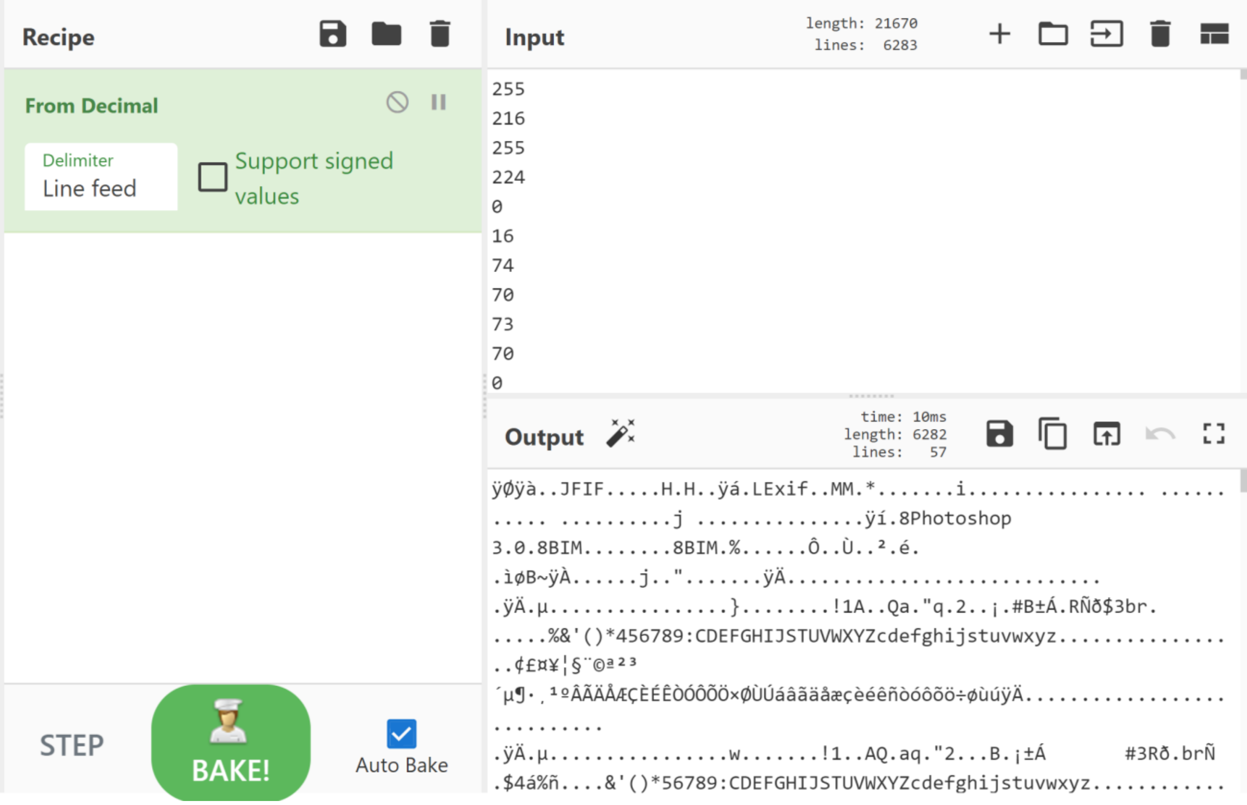


```
root@kali:/home/kali/Pictures# tshark -r sneaky_transmission.pcapng  | grep "(ping) request" | cut -d "=" -f4 | more
Running as user "root" and group "root". This could be dangerous.
255
216
255
224
0
16
74
70
73
70
0
1
1
0
0
72
0
72
```

Get the command works, now let's dump the output into a file.

Command: tshark -r sneaky_transmission.pcapng | grep "(ping) request" | cut -d "=" -f4 > ttl

Looks like we have to convert from decimal, let's use cyberchef https://gchq.github.io/CyberChef/



From Decimal

Delimiter: Line feed

☐ Support signed values

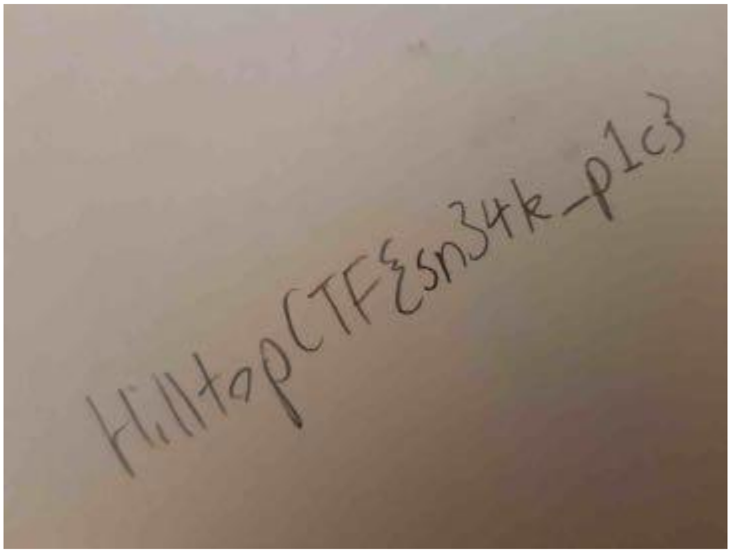
length: 21670
lines: 6283

```
255
216
255
224
0
16
74
70
73
70
0
```

time: 10ms
length: 6282
lines: 57

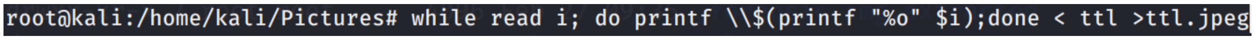
```
ÿØÿà..JFIF.....H.H..ÿá.LExif..MM.*.......i................. ......
..... ..........j ...............ÿí.8Photoshop
3.0.8BIM........8BIM.%......Ô..Ù..².é.
.ìøB~ÿÀ......j.."......ÿÄ..........................
.ÿÄ.µ................}........!1A..Qa."q.2..¡.#B±Á.RÑð$3br.
.....%&'()*456789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz..............
..¢£¤¥¦§¨©ª²³
´µ¶·¸¹ºÂÃÄÅÆÇÈÉÊÒÓÔÕÖ×ØÙÚáâãäåæçèéêñòóôõö÷øùúÿÄ.................
..........
.ÿÄ.µ................w.......!1..AQ.aq."2...B.¡±Á       #3Rð.brÑ
.$4á%ñ....&'()*56789:CDEFGHIJSTUVWXYZcdefghijstuvwxyz...........
```

We can see the JFIF tag, good looks like an image lets download and open it up.



FLAG: HilltopCTF{sn3ek_p1c}

The same can be done in kali with the below command.



```
root@kali:/home/kali/Pictures# while read i; do printf \\$(printf "%o" $i);done < ttl >ttl.jpeg
```