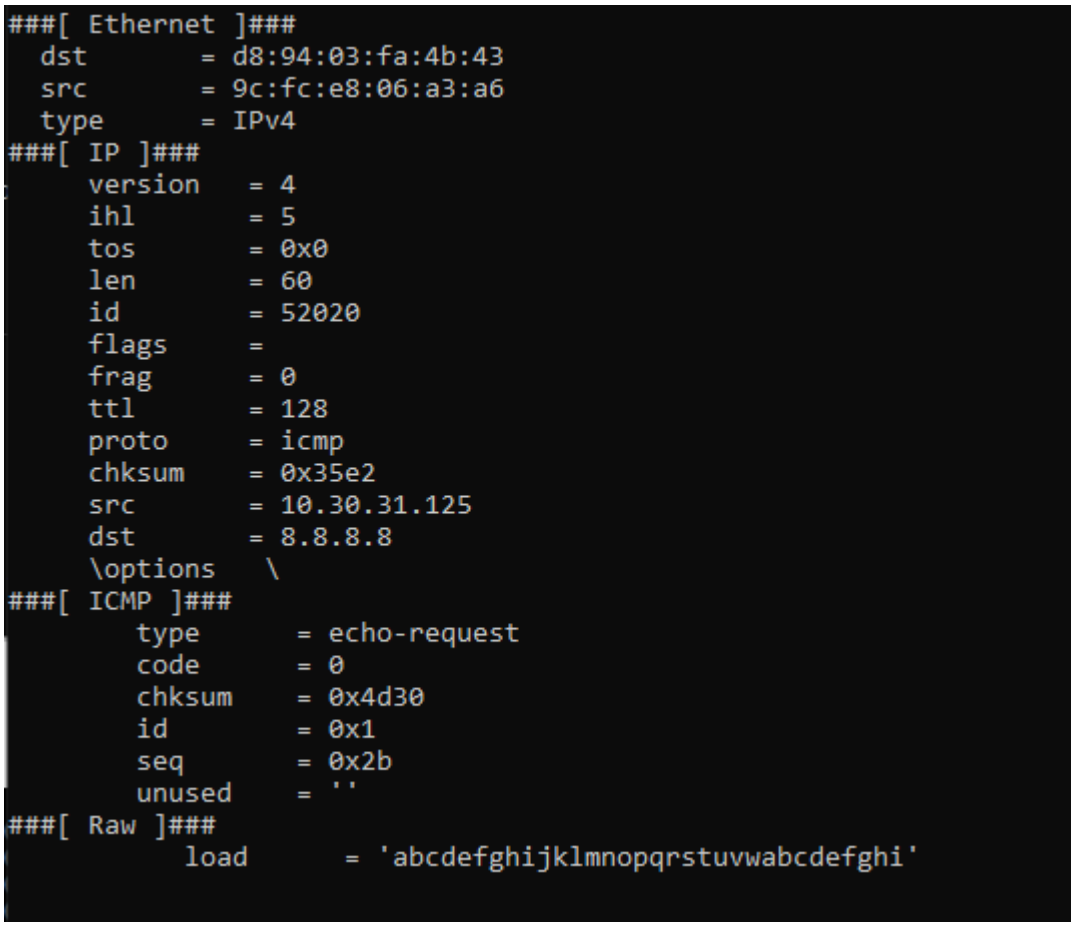
1.

With admin privilege:



Without admin privilege:

```
###[ Ethernet ]###
  dst       = d8:94:03:fa:4b:43
  src       = 9c:fc:e8:06:a3:a6
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 60
     id        = 52019
     flags     =
     frag      = 0
     ttl       = 128
     proto     = icmp
     chksum    = 0x35e3
     src       = 10.30.31.125
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x4d31
        id        = 0x1
        seq       = 0x2a
        unused    = ''
###[ Raw ]###
           load      = 'abcdefghijklmnopqrstuvwabcdefghi'
```

I see no difference on my Windows machine.

1).

```
>>> ls(IP(src='10.0.2.4', dst='www.mit.edu'))
version    : BitField  (4 bits)                = 4               ('4')
ihl        : BitField  (4 bits)                = None            ('None')
tos        : XByteField                        = 0               ('0')
len        : ShortField                        = None            ('None')
id         : ShortField                        = 1               ('1')
flags      : FlagsField                        = <Flag 0 ()>     ('<Flag 0 ()>')
frag       : BitField  (13 bits)               = 0               ('0')
ttl        : ByteField                         = 64              ('64')
proto      : ByteEnumField                     = 0               ('0')
chksum     : XShortField                       = None            ('None')
src        : SourceIPField                     = '10.0.2.4'      ('None')
dst        : DestIPField                       = Net("www.mit.edu/32") ('None')
options    : PacketListField                   = []              ('[]')
>>>
```

2).

```
>>> ls(UDP(sport=5000, dport=53))
sport      : ShortEnumField                    = 5000           ('53')
dport      : ShortEnumField                    = 53             ('53')
len        : ShortField                        = None           ('None')
chksum     : XShortField                       = None           ('None')
>>>
```

```
>>> ls(ICMP())
type       : ByteEnumField                     = 8              ('8')
code       : MultiEnumField (Depends on 8)     = 0              ('0')
chksum     : XShortField                       = None           ('None')
id         : XShortField (Cond)                = 0              ('0')
seq        : XShortField (Cond)                = 0              ('0')
ts_ori     : ICMPTimeStampField (Cond)         = None           ('13348291')
ts_rx      : ICMPTimeStampField (Cond)         = None           ('13348291')
ts_tx      : ICMPTimeStampField (Cond)         = None           ('13348291')
gw         : IPField (Cond)                    = None           ("'0.0.0.0'")
ptr        : ByteField (Cond)                  = None           ('0')
reserved   : ByteField (Cond)                  = None           ('0')
length     : ByteField (Cond)                  = None           ('0')
addr_mask  : IPField (Cond)                    = None           ("'0.0.0.0'")
nexthopmtu : ShortField (Cond)                 = None           ('0')
unused     : MultipleTypeField (ShortField, IntField, StrFixedLenField) = b''             ("b''")
>>>
```

3). stacking IP header over ICMP()

```
>>> (IP(dst="8.8.8.8")/ICMP()).show2()
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 28
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = icmp
  chksum    = 0xbeba
  src       = 172.19.0.3
  dst       = 8.8.8.8
  \options   \
###[ ICMP ]###
     type      = echo-request
     code      = 0
     chksum    = 0xf7ff
     id        = 0x0
     seq       = 0x0
     unused    = ''
```

Similary, UDP()

```
>>> ( IP(src="10.0.2.4", dst='8.8.8.8')/UDP(dport=53) ).show2()
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 28
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = udp
  chksum    = 0x5ebd
  src       = 10.0.2.4
  dst       = 8.8.8.8
  \options   \
###[ UDP ]###
     sport     = domain
     dport     = domain
     len       = 8
     chksum    = 0xe360
```

4).

UDP segment:

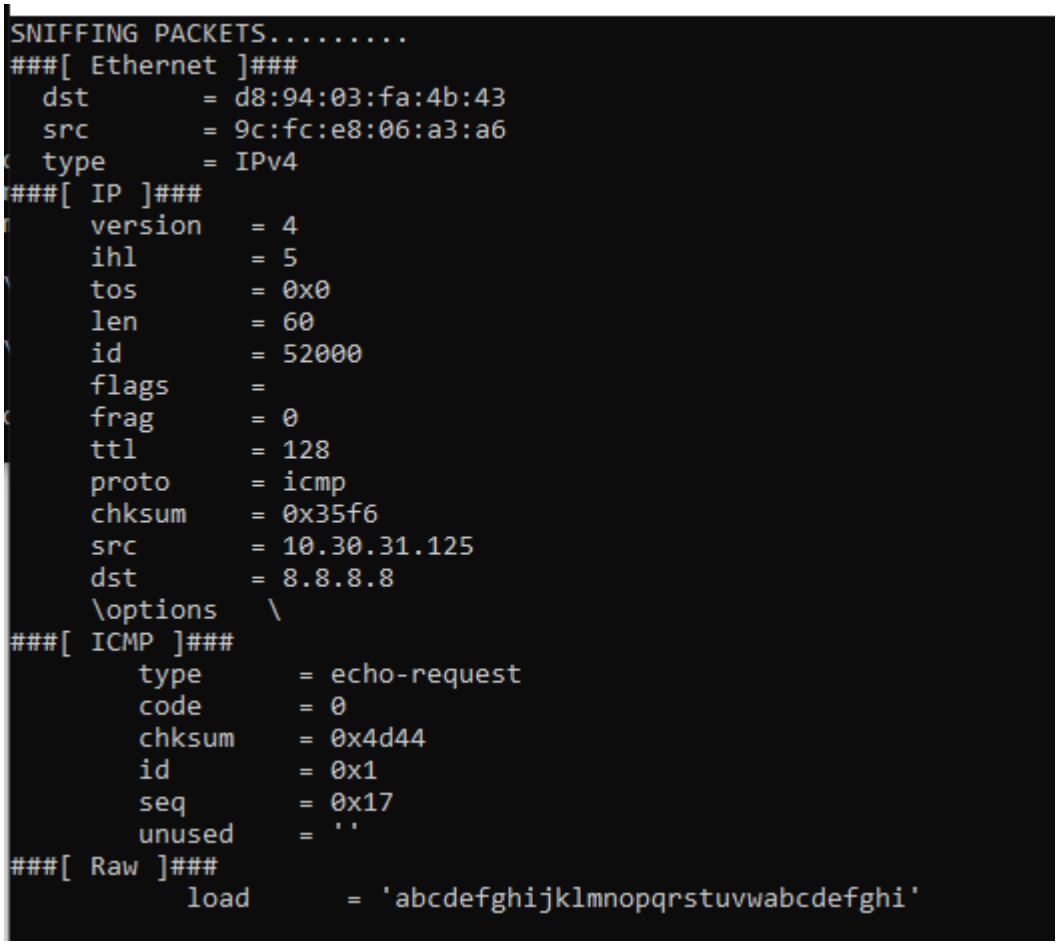
```
>>> ( IP(src="10.0.2.4", dst='8.8.8.8')/UDP(dport=53) )['UDP'].show2()
###[ UDP ]###
  sport     = domain
  dport     = domain
  len       = 8
  chksum    = 0xe360

>>>
```

ICMP segment

```
>>> (IP(dst="8.8.8.8")/ICMP())['ICMP'].show2()
###[ ICMP ]###
  type      = echo-request
  code      = 0
  chksum    = 0xf7ff
  id        = 0x0
  seq       = 0x0
  unused    = ''
```

2. Sniffing Packets

Note, for this lab, I have to use dst instead of pkt[IP].src, as from my machine it is not accessible to 8.8.8.8



3. Sniff function with BPF filters

Applied BPF filters as below:

```
pkt = sniff(filter='icmp and dst 8.8.8.8',prn=print_pkt)
```

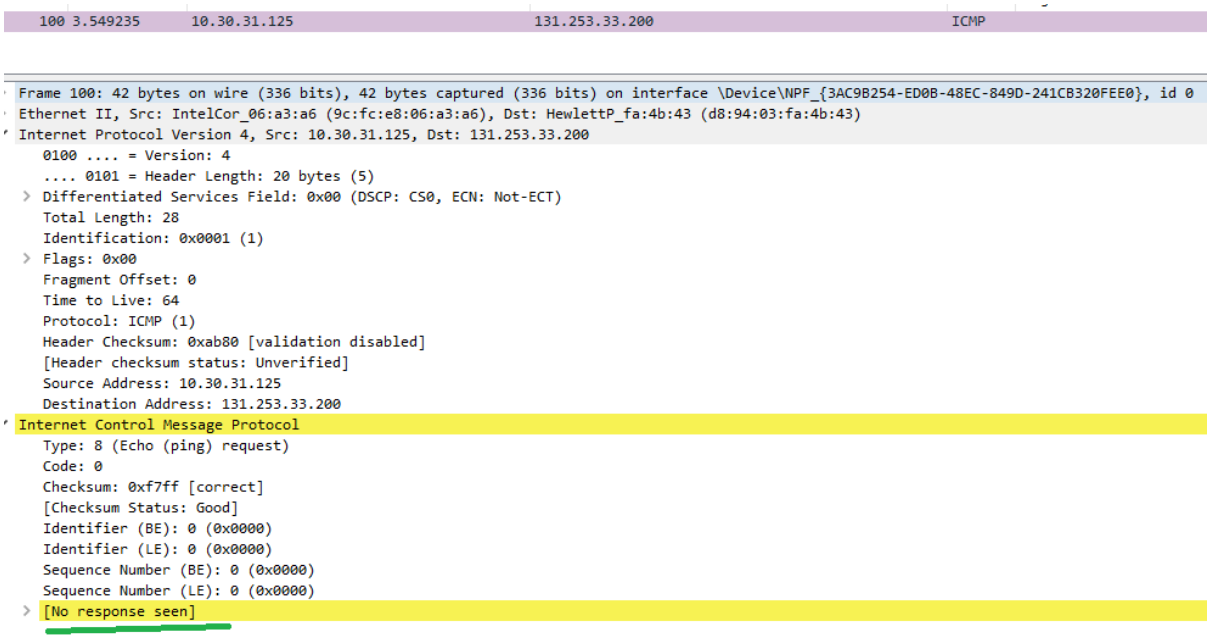
result:

```
###[ Ethernet ]###
  dst       = d8:94:03:fa:4b:43
  src       = 9c:fc:e8:06:a3:a6
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 60
     id        = 52009
     flags     =
     frag      = 0
     ttl       = 128
     proto     = icmp
     chksum    = 0x35ed
     src       = 10.30.31.125
     dst       = 8.8.8.8
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = 0x4d3b
        id        = 0x1
        seq       = 0x20
        unused    = ''
###[ Raw ]###
           load      = 'abcdefghijklmnopqrstuvwabcdefghi'
```

4. Spoofing ICMP Packets

8.8.8.8 is not reachable from my machine, switching to www.bing.com

From Wireshark no response was observed, as a fake src was in the request package, the response probably has been directed to the fake one rather than the real src.



5. sniffing-then-spoofing

8.8.8.8 is not reachable, therefore, using ‘10.5.40.12’ instead. So here are the code executed:

```
#!/usr/bin/python3
from scapy.all import *

def spoof_pkt(pkt):
  if ICMP in pkt and pkt[ICMP].type == 8:

    ip = IP(src=pkt[IP].dst, dst=pkt[IP].src, ihl=pkt[IP].ihl)
    icmp = ICMP(type=0, id=pkt[ICMP].id, seq=pkt[ICMP].seq)
    data = pkt[Raw].load
    newpkt = ip/icmp/data

    print("Spoofed Packet.........")
    print("Source IP : ", newpkt[IP].src)
    print("Destination IP :", newpkt[IP].dst)

    send(newpkt,verbose=0)

pkt = sniff(filter='icmp and host 10.5.40.12',prn=spoof_pkt)
```

and result in Wireshark, I observed there is an unreachable error after everything normal ping-response round.