```
>>> from scapy . all  import *
>>> p = Ether() / IP() / ICMP()
>>> p
<Ether  type=IPv4 |<IP  frag=0 proto=icmp |<ICMP  |>>>
>>>
```

Dans le repertpoire taper sudo python3 puis :
>>> from scapy . all import *
>>> p = Ether () / IP () / ICMP ()
>>> p # Affiche toutes ce dont p est constituer

P.summary()

```
>>> p.summary()
'Ether / IP / ICMP 127.0.0.1 > 127.0.0.1 echo-request 0'
>>>
```

p.show()

```
>>> p.show()
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 00:00:00:00:00:00
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = None
     tos       = 0x0
     len       = None
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = None
     src       = 127.0.0.1
     dst       = 127.0.0.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = None
        id        = 0x0
        seq       = 0x0
        unused    = ''
```

ls(p)

```
>>> ls(p)
dst        : DestMACField                      = 'ff:ff:ff:ff:ff:ff' ('None')
src        : SourceMACField                    = '00:00:00:00:00:00' ('None')
type       : XShortEnumField                   = 2048            ('36864')
--
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                     = 1               ('0')
chksum     : XShortField                       = None            ('None')
src        : SourceIPField                     = '127.0.0.1'     ('None')
dst        : DestIPField                       = '127.0.0.1'     ('None')
options    : PacketListField                   = []              ('[]')
--
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            ('33383375')
ts_rx      : ICMPTimeStampField (Cond)         = None            ('33383375')
ts_tx      : ICMPTimeStampField (Cond)         = None            ('33383375')
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''"
)
```

pour @Mac
dst : 00 : 00 : 00 : 00 : 00 : 00
src : ff : ff : ff : ff : ff : ff

@IP :
src : 127.0.0.1
dst : 127.0.0.1

TTL : 64

ICMP : ICMPTimestampField none

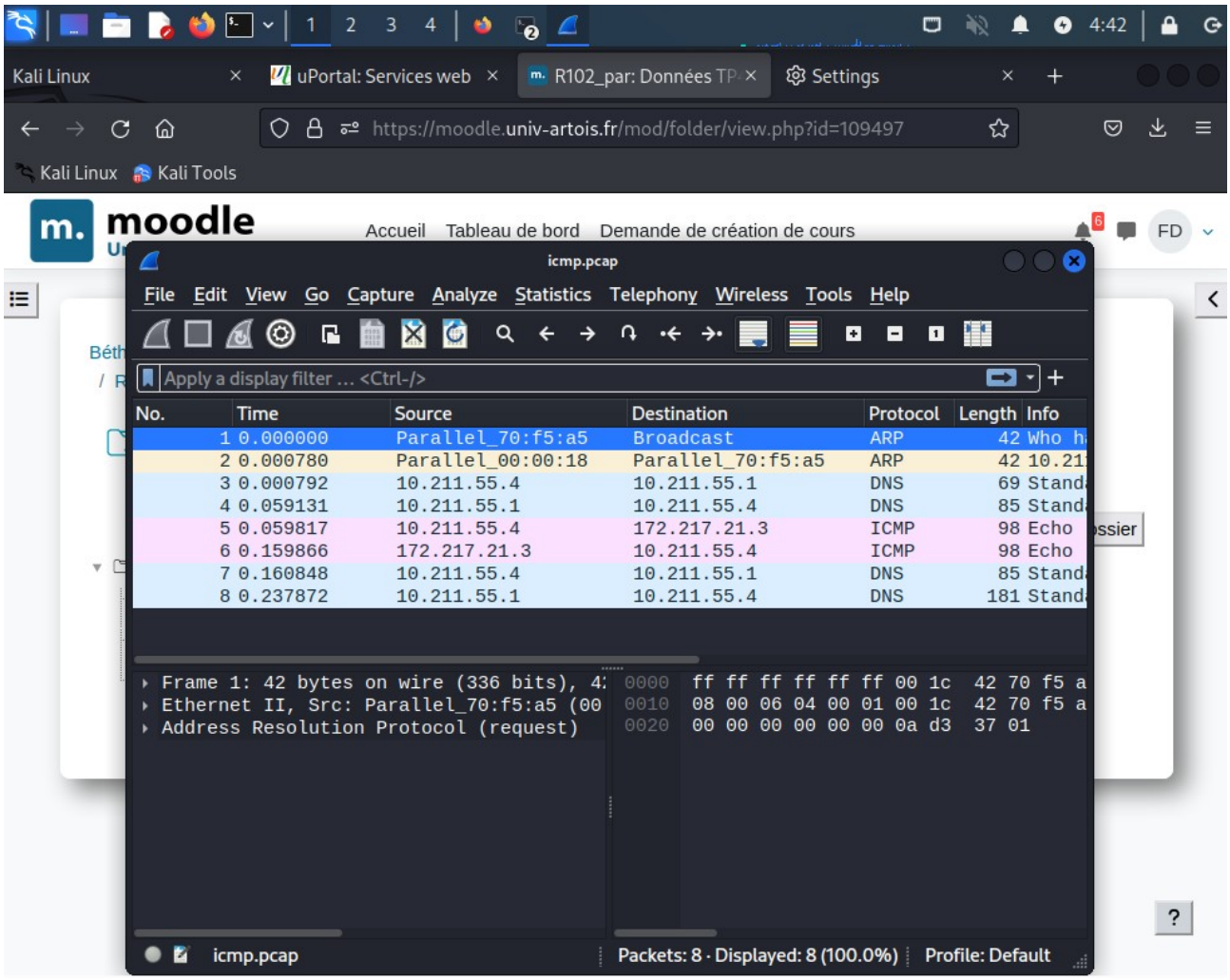
5. la commande hexdump(p)

```
0000  FF FF FF FF FF FF 00 00 00 00 08 00 45 00   ............E.
0010  00 1C 00 01 00 00 40 01 7C DE 7F 00 00 01   ......@.|.....
0020  7F 00 00 01 08 00 F7 FF 00 00 00 00         ............
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
00	00	FF	FF	FF	FF	FF	FF	00	00	00	00	08	00	45	00
E0	02	07	F0	00	00	10	80	0F	7F	F0	00	00	00	00	

6.

7.



>>> p = rdpcap (" icmp . pcap ")
>>> p
>>> p . show ()

8. Examiner la premier trame
dst : 00 : 00 : 00 : 00 : 00 : 00
src : ff : ff : ff : ff : ff : ff

type de protocol : IPv4
version 4
ttl : 64
ipsrc : 127.0.0.1
ipdst : 127.0.0.1

ICMP :
type : echo-request

9.

```
>>> p[0:3].show()
###[ Ethernet ]###
  dst       = ff:ff:ff:ff:ff:ff
  src       = 00:00:00:00:00:00
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = None
     tos       = 0x0
     len       = None
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = None
     src       = 127.0.0.1
     dst       = 127.0.0.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = None
        id        = 0x0
        seq       = 0x0
        unused    = ''
```

10. ping vers la RTBox

>>> p = Ether () / IP (dst = ’192.31.25.1 ’) / ICMP ()

```
>>> p = Ether() / IP(dst ='192.31.25.1') / ICMP()
>>> p.show()
###[ Ethernet ]###
  dst       = 08:00:27:4f:10:ad
  src       = 08:00:27:42:13:7d
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = None
     tos       = 0x0
     len       = None
     id        = 1
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = None
     src       = 192.31.25.12
     dst       = 192.31.25.1
     \options   \
###[ ICMP ]###
        type      = echo-request
        code      = 0
        chksum    = None
        id        = 0x0
        seq       = 0x0
        unused    = ''
```

dst : 08 : 00 : 27 : 4f : 10 : ad

src : 08 : 00 : 27 : 42 : 13 : 7d

type de protocol : IPv4

version 4
ttl : 64
ipsrc : 192.31.25.12
ipdst : 192.31.25.1

ICMP :
type : echo-request

11.

```
>>> p[IP].dst
'192.31.25.1'
>>> p[IP].src
'192.31.25.12'
>>> p[Ether].src
'08:00:27:42:13:7d'
>>> p[Ether].dst
'08:00:27:4f:10:ad'
>>> p [ICMP].type
8
>>>
```

Envoi et reception de datagrammes avec scapy

1. Envoie de paquet

```
>>> p = Ether() / IP(dst='192.31.25.1') / ICMP()
>>> sendp(p)
.
Sent 1 packets.
>>> p = IP(dst='192.31.25.1') / ICMP()
>>> sendp(p)
.
Sent 1 packets.
>>>
```

2. Reponse aux ping

```
>>> srp1(p)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<Ether  dst=08:00:27:42:13:7d src=08:00:27:4f:10:ad type=IPv4 |<IP  version=4 ihl=5 tos=0
x0 len=28 id=20965 flags= frag=0 ttl=64 proto=icmp chksum=0x76b0 src=192.31.25.1 dst=192.
31.25.12 |<ICMP  type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |<Padding  load='\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>>>
>>> srp1(p)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<Ether  dst=08:00:27:42:13:7d src=08:00:27:4f:10:ad type=IPv4 |<IP  version=4 ihl=5 tos=0
x0 len=28 id=47411 flags= frag=0 ttl=64 proto=icmp chksum=0xf62 src=192.31.25.1 dst=192.3
1.25.12 |<ICMP  type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |<Padding  load='\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>>>
```

>>> p = IP (dst = ’192.31.25.1 ’) / ICMP ()

```
>>> srp1(p)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<Ether  dst=08:00:27:42:13:7d src=08:00:27:4f:10:ad type=IPv4 |<IP  version=4 ihl=5 tos=0
x0 len=28 id=20965 flags= frag=0 ttl=64 proto=icmp chksum=0x76b0 src=192.31.25.1 dst=192.
31.25.12 |<ICMP  type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |<Padding  load='\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>>>
>>> srp1(p)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
<Ether  dst=08:00:27:42:13:7d src=08:00:27:4f:10:ad type=IPv4 |<IP  version=4 ihl=5 tos=0
x0 len=28 id=47411 flags= frag=0 ttl=64 proto=icmp chksum=0xf62 src=192.31.25.1 dst=192.3
1.25.12 |<ICMP  type=echo-reply code=0 chksum=0xffff id=0x0 seq=0x0 |<Padding  load='\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00' |>>>>
>>> p = IP(dst='192.31.25.1') / ICMP
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python3/dist-packages/scapy/packet.py", line 576, in __div__
    return other.__rdiv__(self)  # type: ignore
TypeError: Packet.__rdiv__() missing 1 required positional argument: 'other'
>>> p = IP(dst='192.31.25.1') / ICMP()
>>> sr1p (p)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sr1p' is not defined
>>> p = IP(dst='192.31.25.1') / ICMP()
```

>>> sr1 (p)

```
>>> rep = srp1(p)
Begin emission:
Finished sending 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
>>>
```

```
>>> rep.summary()
'Ether / IP / ICMP 192.31.25.1 > 192.31.25.12 echo-reply 0 / Padding'
```

```
>>> rep.summary()
'Ether / IP / ICMP 192.31.25.1 > 192.31.25.12 echo-reply 0 / Padding'
>>> rep.show()
###[ Ethernet ]###
  dst       = 08:00:27:42:13:7d
  src       = 08:00:27:4f:10:ad
  type      = IPv4
###[ IP ]###
     version   = 4
     ihl       = 5
     tos       = 0x0
     len       = 28
     id        = 43053
     flags     =
     frag      = 0
     ttl       = 64
     proto     = icmp
     chksum    = 0x2068
     src       = 192.31.25.1
     dst       = 192.31.25.12
     \options   \
###[ ICMP ]###
        type      = echo-reply
        code      = 0
        chksum    = 0xffff
        id        = 0x0
        seq       = 0x0
        unused    = ''
###[ Padding ]###
           load      = '\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00'

>>>
```

3.
>>> p = IP (dst =" @IP_d ’ une_machine_non_accessible ") / ICMP ()
>>> rep = sr1 (p , timeout =5)
>>> print (rep) # Renvoie none

Script scan pings

Requête Arp

DNS :

Requête DNS

1. Rappeler l’encapsulation du protocole DNS dans un réseau filaire Ethernet ?

ETH | IP | UDP | DNS

2 :

```
┌──(kali㉿kali)-[~/R102TP4]
└─$ nslookup rt-bethune.univ-artois.fr
;; communications error to 172.18.26.101#53: timed out
Server:         172.18.26.101
Address:        172.18.26.101#53

Non-authoritative answer:
Name:   rt-bethune.univ-artois.fr
Address: 193.49.114.18
```

3:

```
┌──(kali㉿kali)-[~/R102TP4]
└─$ sudo python3
Python 3.10.8 (main, Nov  4 2022, 09:21:25) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from scapy.all import *
>>> p = IP(dst='172.18.26.101')/UDP()/DNS(qd=DNSQR(qname='rt-bethune.univ-
artois.fr'))
>>> rep = sr1(p)
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
>>> rep.summary
<bound method Packet.summary of <IP  version=4 ihl=5 tos=0x0 len=87 id=253
04 flags= frag=0 ttl=126 proto=udp chksum=0x3a1b src=172.18.26.101 dst=192
.31.25.12 |<UDP  sport=domain dport=domain len=67 chksum=0xee26 |<DNS  id=
0 qr=1 opcode=QUERY aa=0 tc=0 rd=1 ra=1 z=0 ad=0 cd=0 rcode=ok qdcount=1 a
ncount=1 nscount=0 arcount=0 qd=<DNSQR  qname='rt-bethune.univ-artois.fr.'
 qtype=A qclass=IN ▷ an=<DNSRR  rrname='rt-bethune.univ-artois.fr.' type=
A rclass=IN ttl=81566 rdlen=None rdata=193.49.114.18 ▷ ns=None ar=None ▷
>>>
>>> rep.show
<bound method Packet.show of <IP  version=4 ihl=5 tos=0x0 len=87 id=25304
flags= frag=0 ttl=126 proto=udp chksum=0x3a1b src=172.18.26.101 dst=192.31
.25.12 |<UDP  sport=domain dport=domain len=67 chksum=0xee26 |<DNS  id=0 q
r=1 opcode=QUERY aa=0 tc=0 rd=1 ra=1 z=0 ad=0 cd=0 rcode=ok qdcount=1 anco
unt=1 nscount=0 arcount=0 qd=<DNSQR  qname='rt-bethune.univ-artois.fr.' qt
ype=A qclass=IN ▷ an=<DNSRR  rrname='rt-bethune.univ-artois.fr.' type=A r
class=IN ttl=81566 rdlen=None rdata=193.49.114.18 ▷ ns=None ar=None |>>>>
```

SCRIPT SCAN DE PORT :

1. Pour commencer, vous allez tester les demandes de connexion TCP qui se font avec un segment TCP avec un drapeau/bit Syn (S) activé. Si le service est actif il commencera par répondre par un segment TCP avec les drapeaux/bits SYN/ACK (SA) activés. Testez sous l'outil interactif python3 les instructions suivantes :

```
>>> p = IP(dst='@IP_passerelle_RTBox2')/TCP(dport=22,
   flags='S')
>>> rep = sr1(p,timeout=1)
>>> rep.show()
```

```
>>> p = IP(dst='192.31.25.1')/TCP(dport=22,flags='S')
>>> rep = sr1(p,timeout=1)
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
>>> rep.show
<bound method Packet.show of <IP  version=4 ihl=5 tos=0x10 len=44 id=0 fla
gs=DF frag=0 ttl=64 proto=tcp chksum=0x8870 src=192.31.25.1 dst=192.31.25.
12 |<TCP  sport=ssh dport=ftp_data seq=631336144 ack=1 dataofs=6 reserved=
0 flags=SA window=29200 chksum=0xe11d urgptr=0 options=[('MSS', 1460)] |<P
adding  load='\x00\x00' |>>>>
```

2.

```
┌──(kali㉿kali)-[~/R102TP4]
└─$ sudo ./scanports 192.168.2.1
sudo: ./scanports: command not found

┌──(kali㉿kali)-[~/R102TP4]
└─$ ls
icmp.pcap  p.pcap  scanpings.py  scanports.py

┌──(kali㉿kali)-[~/R102TP4]
└─$ chmod u+x scanports.py

┌──(kali㉿kali)-[~/R102TP4]
└─$ sudo ./scanports.py 192.31.25.1
Port 22 ouvert sur 192.31.25.1
Port 80 ouvert sur 192.31.25.1
```