University of Colorado Boulder

# Network Management and Operations
# TLEN 5410

Python Library - Scapy

**Levi Perigo, Ph.D.**
**University of Colorado Boulder**
**Interdisciplinary Telecom Program**

## Testing - Scapy

- **Packet manipulation tool**
- **Automated tools do not fully work with every system**
- **Highly Configurable**
- **Python Extension**

# Using Scapy

- **Start scapy in shell (must run as root)**
  - sudo scapy

- **Commands are run from within Scapy**

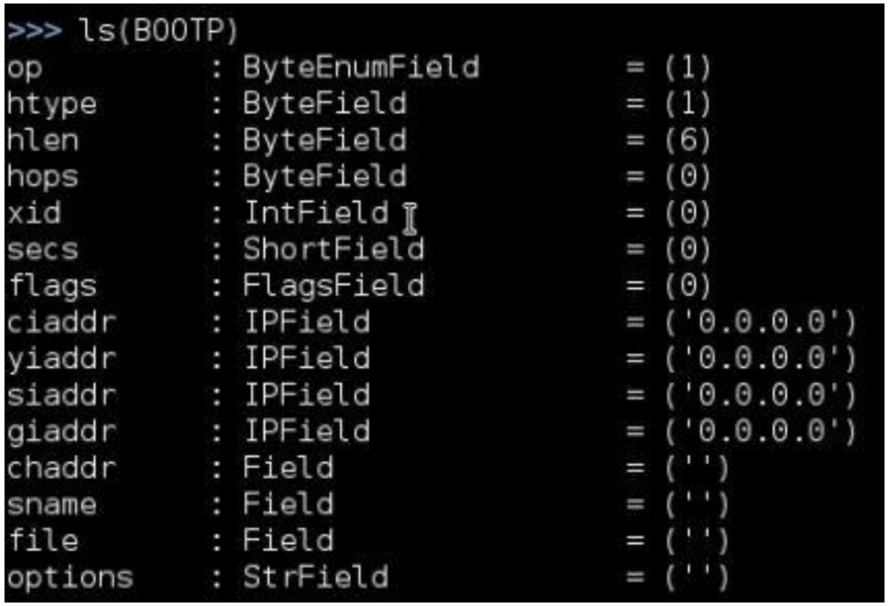
- **.py file - Make sure program runs as root!**

- **Make sure NIC is in promiscuous mode**
  - "allow all" in VirtualBox settings
  - Might be sending traffic with incorrect MAC/IP
    - ***Make sure you see the results***

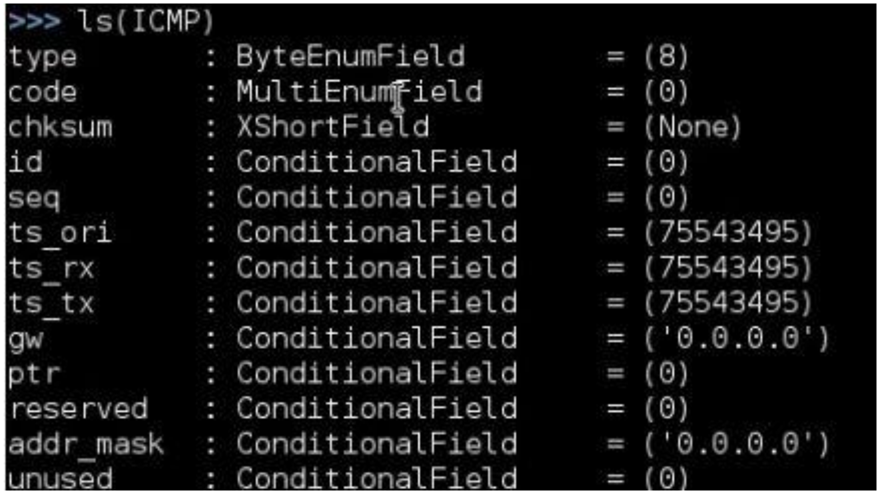
- **lsc()**
  - Function in the scapy interpreter
    - ***Shows the list of <u>commands</u>***

# Scapy

- **ls()**
  - List of all the <u>protocols</u> available in scapy (default values)



```
>>> ls(BOOTP)
op          : ByteEnumField        = (1)
htype       : ByteField            = (1)
hlen        : ByteField            = (6)
hops        : ByteField            = (0)
xid         : IntField             = (0)
secs        : ShortField           = (0)
flags       : FlagsField           = (0)
ciaddr      : IPField              = ('0.0.0.0')
yiaddr      : IPField              = ('0.0.0.0')
siaddr      : IPField              = ('0.0.0.0')
giaddr      : IPField              = ('0.0.0.0')
chaddr      : Field                = ('')
sname       : Field                = ('')
file        : Field                = ('')
options     : StrField             = ('')
```



```
>>> ls(ICMP)
type        : ByteEnumField        = (8)
code        : MultiEnumField       = (0)
chksum      : XShortField          = (None)
id          : ConditionalField     = (0)
seq         : ConditionalField     = (0)
ts_ori      : ConditionalField     = (75543495)
ts_rx       : ConditionalField     = (75543495)
ts_tx       : ConditionalField     = (75543495)
gw          : ConditionalField     = ('0.0.0.0')
ptr         : ConditionalField     = (0)
reserved    : ConditionalField     = (0)
addr_mask   : ConditionalField     = ('0.0.0.0')
unused      : ConditionalField     = (0)
```

## Build Your First Packet

```
>a=IP(ttl=10)
>a
<IP  ttl=10 |>

> a.src='127.0.0.1’
> a.dst="10.0.2.15”
> a
<IP  ttl=10 src= 127.0.0.1 dst=10.0.2.15 |>

>del(a.ttl)
> a
<IP src= 127.0.0.1 dst=10.0.2.15 |>
```

## Layers

```
> IP()
<IP  |>

> IP()/TCP()
<IP  frag=0 proto=tcp |<TCP  |>

> Ether()/IP()/TCP()
<Ether  type=0x800 |<IP  frag=0 proto=tcp |<TCP
```

## Sending Packets

- **send()**
  - Sends packets at layer 3

- **send(IP(dst=“1.2.3.4”)/ICMP())**
  - Sends one packet to IP address 1.2.3.4 using ICMP

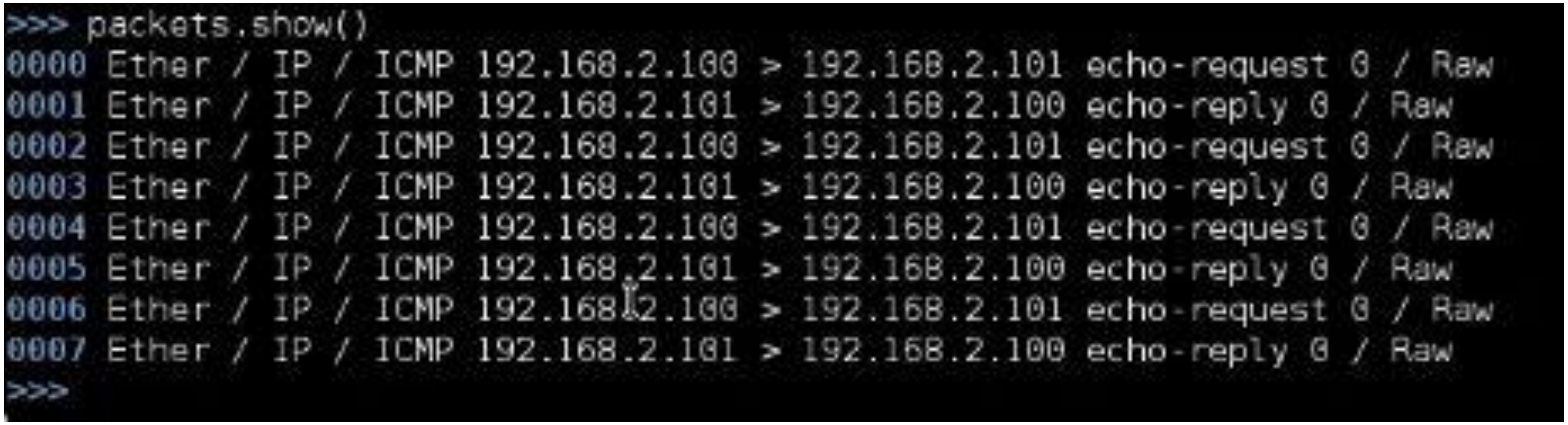
## Example

```
>>test = IP(src="10.1.1.1",dst="10.1.1.2",ttl=(1,4))/UDP(dport=67)

>>send(test)
```

## Sniff Packets

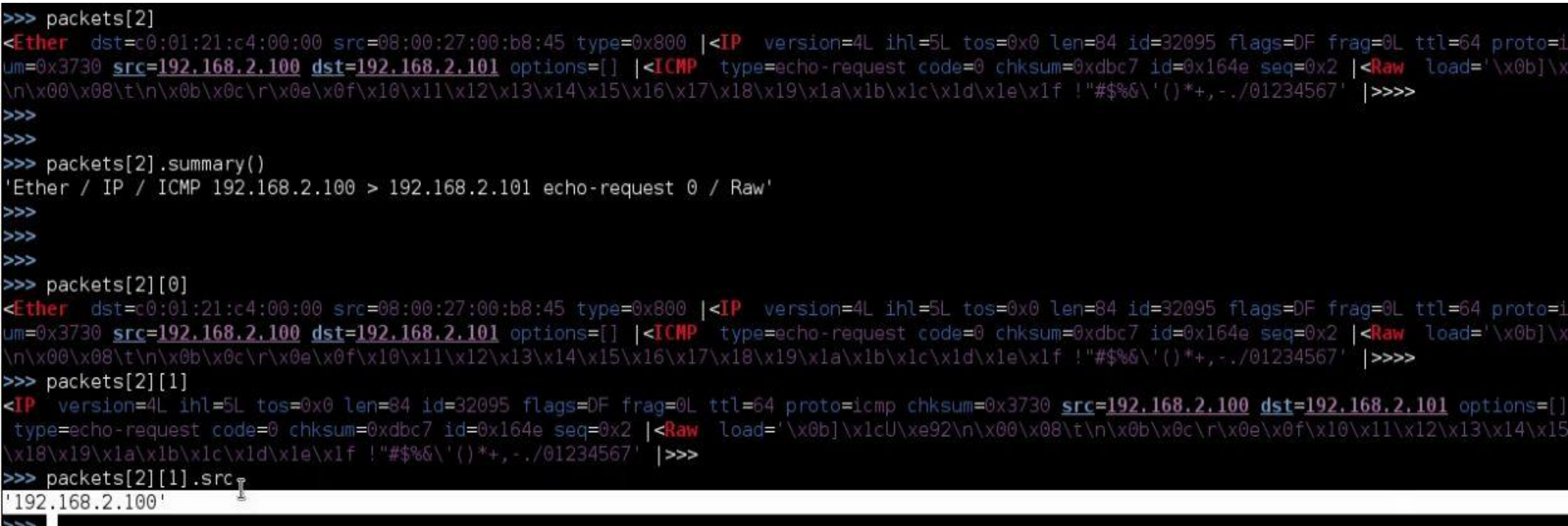
- >>> packets = sniff(filter="ICMP", iface="eth1")
- "Ping from the machine capturing"
- packets.show() – (shows captured packets)



```
>>> packets.show()
0000 Ether / IP / ICMP 192.168.2.100 > 192.168.2.101 echo-request 0 / Raw
0001 Ether / IP / ICMP 192.168.2.101 > 192.168.2.100 echo-reply 0 / Raw
0002 Ether / IP / ICMP 192.168.2.100 > 192.168.2.101 echo-request 0 / Raw
0003 Ether / IP / ICMP 192.168.2.101 > 192.168.2.100 echo-reply 0 / Raw
0004 Ether / IP / ICMP 192.168.2.100 > 192.168.2.101 echo-request 0 / Raw
0005 Ether / IP / ICMP 192.168.2.101 > 192.168.2.100 echo-reply 0 / Raw
0006 Ether / IP / ICMP 192.168.2.100 > 192.168.2.101 echo-request 0 / Raw
0007 Ether / IP / ICMP 192.168.2.101 > 192.168.2.100 echo-reply 0 / Raw
>>>
```

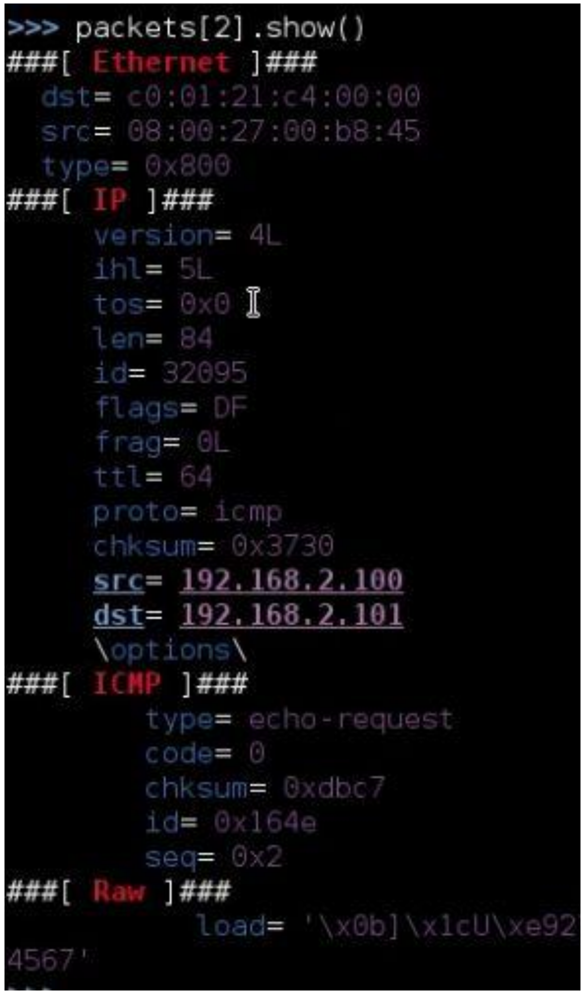
## Sniff Packets

– To view packets use element in list (“packet number”) :
  - ***packets[2]***
  - ***Also helpful to use “summary”***
    – packets[2].summary()



```
>>> packets[2]
<Ether  dst=c0:01:21:c4:00:00 src=08:00:27:00:b8:45 type=0x800 |<IP  version=4L ihl=5L tos=0x0 len=84 id=32095 flags=DF frag=0L ttl=64 proto=i
um=0x3730 src=192.168.2.100 dst=192.168.2.101 options=[] |<ICMP  type=echo-request code=0 chksum=0xdbc7 id=0x164e seq=0x2 |<Raw  load='\x0b]\x
\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567' |>>>>
>>>
>>>
>>> packets[2].summary()
'Ether / IP / ICMP 192.168.2.100 > 192.168.2.101 echo-request 0 / Raw'
>>>
>>>
>>>
>>> packets[2][0]
<Ether  dst=c0:01:21:c4:00:00 src=08:00:27:00:b8:45 type=0x800 |<IP  version=4L ihl=5L tos=0x0 len=84 id=32095 flags=DF frag=0L ttl=64 proto=i
um=0x3730 src=192.168.2.100 dst=192.168.2.101 options=[] |<ICMP  type=echo-request code=0 chksum=0xdbc7 id=0x164e seq=0x2 |<Raw  load='\x0b]\x
\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567' |>>>>
>>> packets[2][1]
<IP  version=4L ihl=5L tos=0x0 len=84 id=32095 flags=DF frag=0L ttl=64 proto=icmp chksum=0x3730 src=192.168.2.100 dst=192.168.2.101 options=[]
 type=echo-request code=0 chksum=0xdbc7 id=0x164e seq=0x2 |<Raw  load='\x0b]\x1cU\xe92\n\x00\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15
\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f !"#$%&\'()*+,-./01234567' |>>>
>>> packets[2][1].src
'192.168.2.100'
```

# Sniff & Show (summary)



```
>>> packets[2].show()
###[ Ethernet ]###
  dst= c0:01:21:c4:00:00
  src= 08:00:27:00:b8:45
  type= 0x800
###[ IP ]###
     version= 4L
     ihl= 5L
     tos= 0x0
     len= 84
     id= 32095
     flags= DF
     frag= 0L
     ttl= 64
     proto= icmp
     chksum= 0x3730
     src= 192.168.2.100
     dst= 192.168.2.101
     \options\
###[ ICMP ]###
        type= echo-request
        code= 0
        chksum= 0xdbc7
        id= 0x164e
        seq= 0x2
###[ Raw ]###
           load= '\x0b]\x1cU\xe92
4567'
```

University of Colorado
Boulder

# Scapy Notes - Summary

- See a list of what commands Scapy has available:  lsc()

- See ALL the supported protocols: ls()

- See the fields and default values for any protocol: ls("*protocol*")

- See packet layers: .summary()

- See packet contents: .show()

- Dig into a specific packet layer using a list index: pkts[3][2].summary()...
    - ***the first index chooses the packet out of the pkts list (from the .show() function, the second index chooses the layer for that specific packet.***

- Return a string of the command necessary to recreate that sniffed packet: .command()