

# Covering Packet Manipulation with Scapy

---



**Sean Wilkins**

Network Engineer & Author

[swilkins@infodispersion.com](mailto:swilkins@infodispersion.com)

[www.infodispersion.com](http://www.infodispersion.com)



# Module Introduction

**Sniffing knowledge is  
important**

**Packet creation and  
dissection is just as important**



# Overview



- **Reviewing Common Scapy Packet Display Commands and Methods**
- **Concepts Demonstration - Display Commands**
- **Methods of Creating and Dissecting Traffic**
- **Concepts Demonstration - Creating and Dissecting Traffic**



This module is based on a  
scenario covered previously



# Ready to Move On?

**Extract information  
from already  
captured packets**

**Build new packets  
with this  
information**

**Build packets to  
perform a specific  
task**





**Need to be able to analyze networking traffic**

**Can be done through security appliances**

**Sometimes things slip through**

**Python and Scapy are useful in these situations**



**Scapy can display  
previously collected  
information**

**Commands include:**

- **show**
- **summary**
- **nsummary**
- **hexdump**
  - **filter**
  - **ls**
- **show2**
- **sprintf**



# .show command

```
>>> a[0].show()
###[ Ethernet ]###
  dst      = 94:bf:94:a0:d4:b4
  src      = 1c:1b:0d:0f:71:e4
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 61
  id       = 64911
  flags    = DF
  frag     = 0
  ttl      = 128
  proto    = udp
  chksum   = 0x4072
  src      = 192.168.1.95
  dst      = 64.233.185.189
  \options \
###[ UDP ]###
  sport    = 56450
  dport    = https
  len      = 41
  chksum   = 0x55ee
###[ Raw ]###
  load     = 'L\\xe5\\xa1\\xd95\\xad\\xcbNVk\\xc8\\xc4L\\x97\\xd7\\xec1+\\xec8-
\\xc78m\\x12\\xbf%2%\\xb1\\xef\\x16\\xd0'
```



# .summary Command

```
>>> a.summary()
Ether / IP / UDP 192.168.1.95:56450 > 64.233.185.189:https / Raw
Ether / ARP who has 192.168.1.181 says 192.168.1.181 / Padding
Ether / ARP who has 192.168.1.180 says 192.168.1.180 / Padding
Ether / IP / UDP / DNS Ans "b'Android-4.local.'"
Ether / IPv6 / UDP / DNS Ans "b'Android-4.local.'"
Ether / IP / UDP / DNS Qry "b'126.1.168.192.in-addr.arpa.'"
Ether / IP / UDP / DNS Qry "b'126.1.168.192.in-addr.arpa.'"
Ether / IP / UDP 192.168.1.107:10102 > 192.168.1.255:10102 / Raw
Ether / IP / TCP 192.168.1.95:1686 > 192.168.1.117:32052 PA / Raw
Ether / IP / TCP 192.168.1.95:1685 > 192.168.1.117:8009 PA / Raw
Ether / IP / TCP 162.247.241.14:https > 192.168.1.95:1802 A / Padding
Ether / IP / TCP 192.168.1.95:1802 > 162.247.241.14:https A / Padding
Ether / IP / TCP 192.168.1.117:8009 > 192.168.1.95:1685 PA / Raw
Ether / IP / TCP 192.168.1.117:32052 > 192.168.1.95:1686 PA / Raw
Ether / IP / TCP 192.168.1.95:1686 > 192.168.1.117:32052 A / Padding
Ether / IP / TCP 192.168.1.95:1685 > 192.168.1.117:8009 A / Padding
Ether / IP / UDP / DNS Qry "b'126.1.168.192.in-addr.arpa.'"
Ether / IP / TCP 162.247.241.14:https > 192.168.1.95:1545 A / Padding
Ether / IP / TCP 192.168.1.95:1545 > 162.247.241.14:https A / Padding
Ether / IP / UDP 192.168.1.181:49154 > 255.255.255.255:6667 / Raw
>>> a[0].summary()
'Ether / IP / UDP 192.168.1.95:56450 > 64.233.185.189:https / Raw'
>>>
```



# .hexdump Command

```
>>> a.hexdump()
0000 01:00:48.197745 Ether / IP / UDP 192.168.1.95:56450 > 64.233.185.189:https / Raw
0000  94 BF 94 A0 D4 B4 1C 1B 0D 0F 71 E4 08 00 45 00  .....q...E.
0010  00 3D FD 8F 40 00 80 11 40 72 C0 A8 01 5F 40 E9  .=...@...@r..._@.
0020  B9 BD DC 82 01 BB 00 29 55 EE 4C E5 A1 D9 35 AD  .....)U.L...5.
0030  CB 4E 56 6B C8 C4 4C 97 D7 EC 6C 2B EC 38 2D C7  .NVk..L...l+.8-.
0040  38 6D 12 BF 25 32 25 B1 EF 16 D0                8m..%2%.....
0001 01:00:48.200824 Ether / ARP who has 192.168.1.181 says 192.168.1.181 / Padding
0000  FF FF FF FF FF FF 10 52 1C F7 0E D5 08 06 00 01  .....R.....
0010  08 00 06 04 00 01 10 52 1C F7 0E D5 C0 A8 01 B5  .....R.....
0020  00 00 00 00 00 00 C0 A8 01 B5 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0002 01:00:48.256826 Ether / ARP who has 192.168.1.180 says 192.168.1.180 / Padding
0000  FF FF FF FF FF FF 40 F5 20 09 F9 9D 08 06 00 01  .....@. ....
0010  08 00 06 04 00 01 40 F5 20 09 F9 9D C0 A8 01 B4  .....@. ....
0020  00 00 00 00 00 00 C0 A8 01 B4 00 00 00 00 00 00  .....
0030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
...
```



# .filter Command

```
>>> a
<Sniffed: TCP:10 UDP:8 ICMP:0
Other:2>
>>> a.filter(lambda s:UDP in s)
<filtered Sniffed: TCP:0 UDP:8
ICMP:0 Other:0>
>>> a.filter(lambda s:TCP in s)
<filtered Sniffed: TCP:10 UDP:0
ICMP:0 Other:0>
>>>
```



```

>>> ls(a[0])
dst          : DestMACField          = '94:bf:94:a0:d4:b4' ('None')
src          : SourceMACField        = '1c:1b:0d:0f:71:e4' ('None')
type         : XShortEnumField      = 2048                ('36864')
--
version      : BitField (4 bits)     = 4                   ('4')
ihl          : BitField (4 bits)     = 5                   ('None')
tos          : XByteField            = 0                   ('0')
len          : ShortField            = 61                  ('None')
id           : ShortField            = 64911               ('1')
flags        : FlagsField           = <Flag 2 (DF)>      ('<Flag 0 ()>')
frag         : BitField (13 bits)    = 0                   ('0')
ttl          : ByteField             = 128                 ('64')
proto        : ByteEnumField         = 17                  ('0')
chksum       : XShortField           = 16498               ('None')
src          : SourceIPField         = '192.168.1.95'     ('None')
dst          : DestIPField           = '64.233.185.189'   ('None')
options      : PacketListField      = []                  ('[]')
--
sport        : ShortEnumField        = 56450               ('53')
dport        : ShortEnumField        = 443                 ('53')
len          : ShortField            = 41                  ('None')
chksum       : XShortField           = 21998               ('None')
--
load         : StrField              =
b'L\xe5\xa1\xd95\xad\xcbNVk\xc8\xc4L\x97\xd7\xecl+\xec8-\xc78m\x12\xbf%2%\xb1\xef\x16\xd0' ("b' ")

```



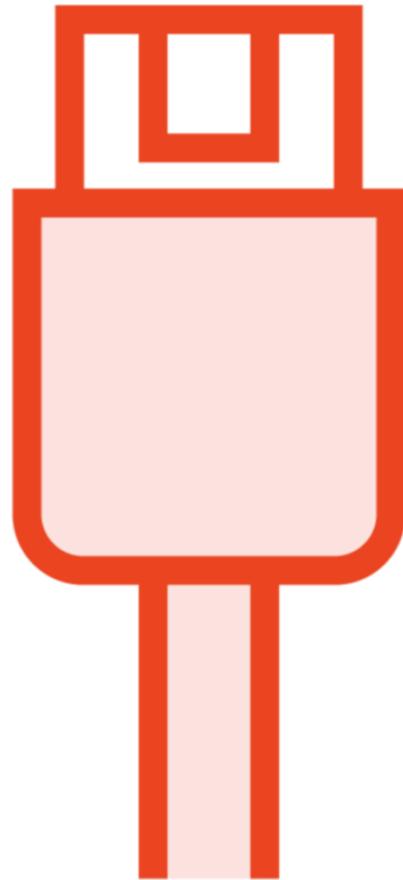
```
>>> a[0].show2()
###[ Ethernet ]###
  dst      = 94:bf:94:a0:d4:b4
  src      = 1c:1b:0d:0f:71:e4
  type     = IPv4
###[ IP ]###
  version  = 4
  ihl      = 5
  tos      = 0x0
  len      = 61
  id       = 64911
  flags    = DF
  frag     = 0
  ttl      = 128
  proto    = udp
  chksum   = 0x4072
  src      = 192.168.1.95
  dst      = 64.233.185.189
  \options \
###[ UDP ]###
  sport    = 56450
  dport    = https
  len      = 41
  chksum   = 0x55ee
###[ Raw ]###
  load     = 'L\\xe5\\xa1\\xd95\\xad\\xcbNVk\\xc8\\xc4L\\x97\\xd7\\xec1+\\xec8-
\\xc78m\\x12\\xbf%2%\\xb1\\xef\\x16\\xd0'
```



# .sprintf Command

```
[ans is a send and receive list  
>>> ans.summary(lambda s,r:s sprintf("%TCP.sport%"))  
43563  
25984  
54534  
30570  
54364  
>>>  
]
```





## Referencing packets inside variable types

### Use packet methods directly

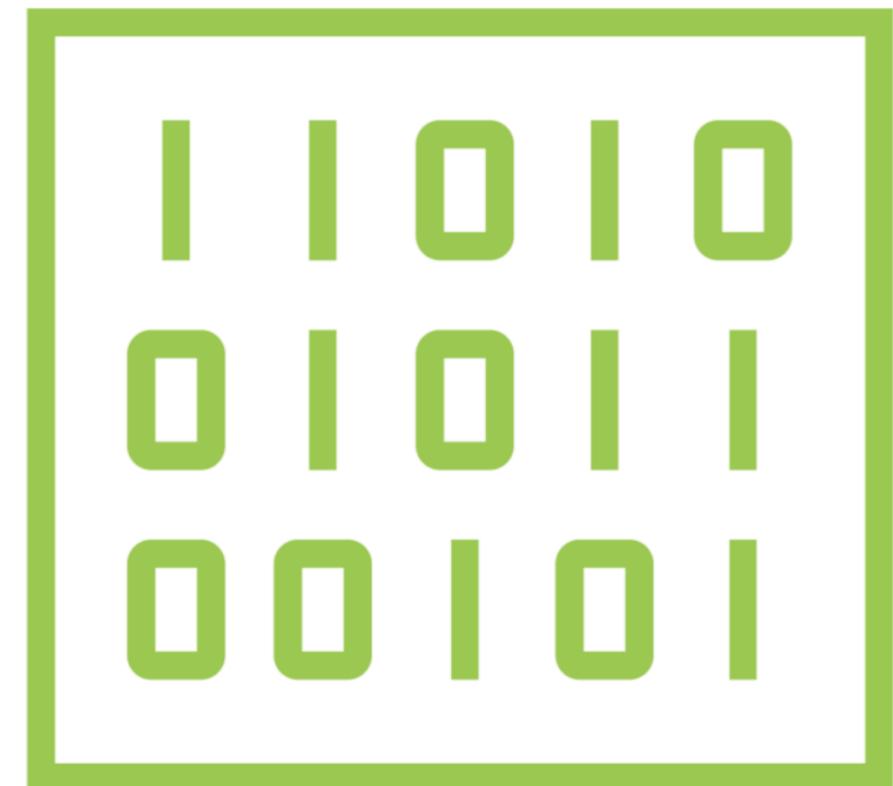
**a = Ether packet type**

- **a.show()**
  - Shows the contents of a packet
- **ls(a)**
  - Shows the contents but in a different format

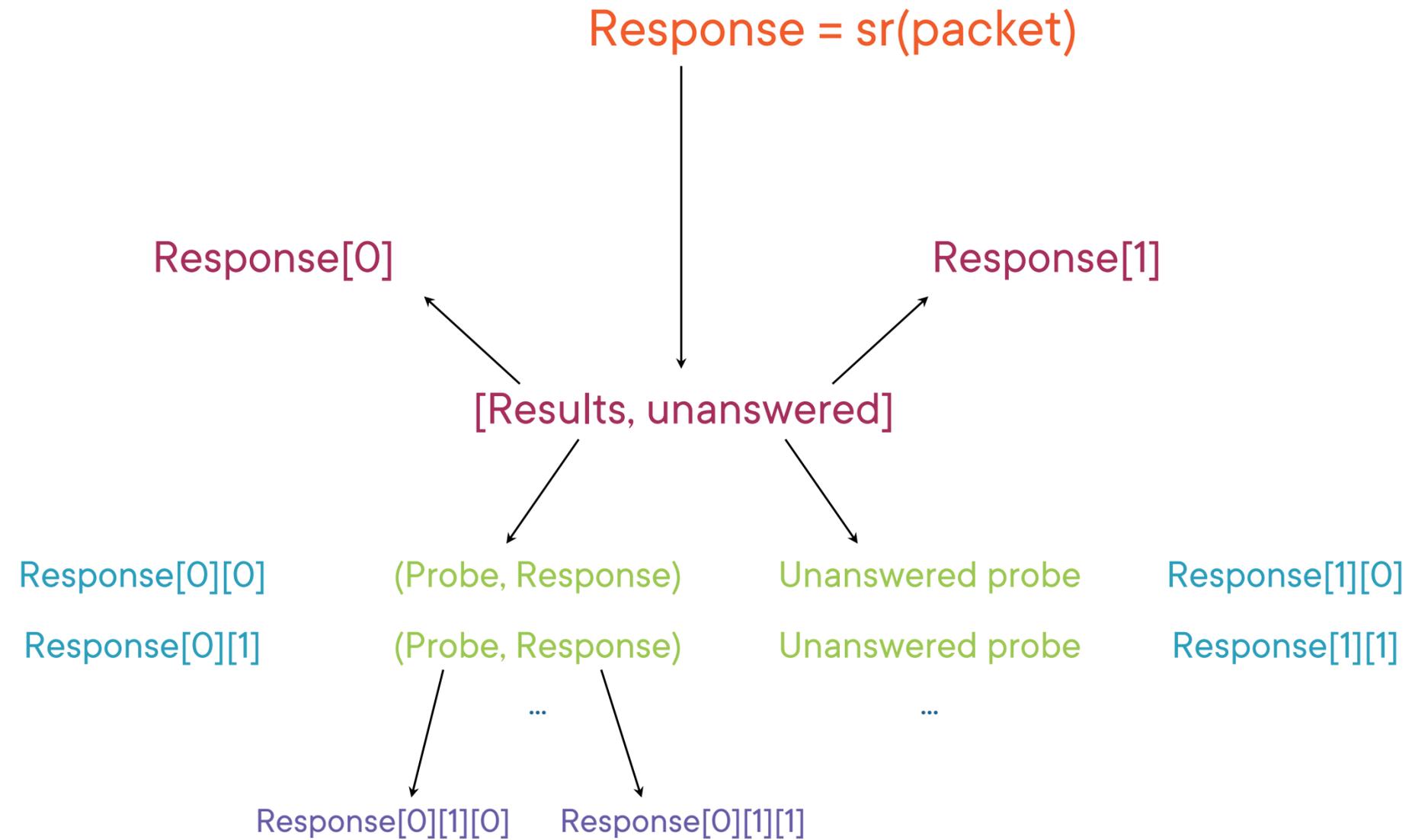


# Packet Reference Options

**a = PacketList**  
**Individual packets can be referenced**  
**a[0] = first packet**  
**a[1] = second packet**



# Packet Reference Options





# wireshark command

{y, x}

**Well respected**

**Free sniffing GUI**

**Scapy provides a view of the variable contents**



# How is this knowledge helpful?

**Need to be able to capture traffic**

**Important to be able to display information**

**Can isolate an attack**

**Determine what the attack is trying to do**

**Test hypothesis and then block the attack**



Let's discuss the creation and dissection of individual packets and their fields.



**Python and Socket can send and receive traffic**

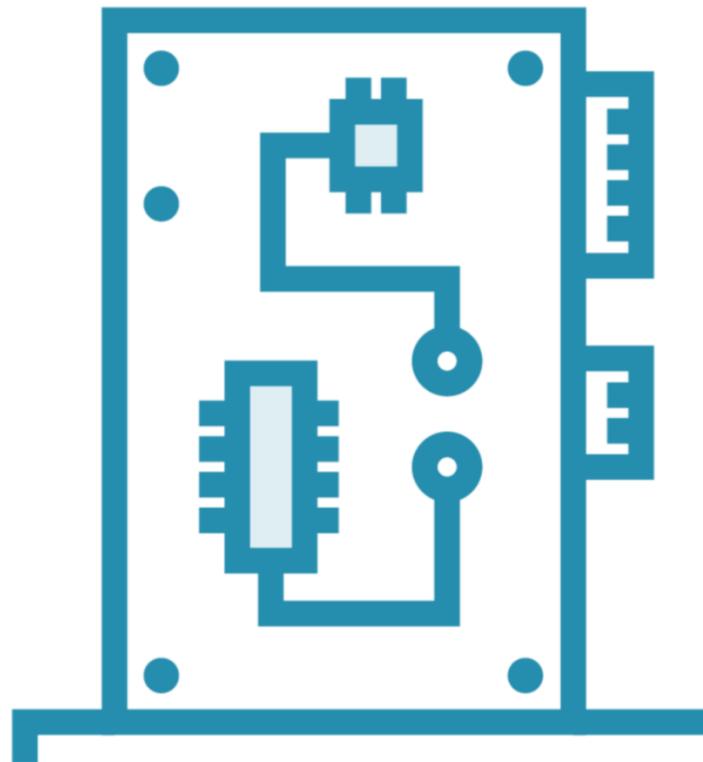
**Socket module requires raw packet processing**

**Scapy has this built into their libraries**

**Choosing Scapy or Socket (or both) depends  
on needs**



# Review



## What makes up the different parts of a packet

- Ethernet
- IP
- TCP
- UDP
- ICMP

## How to build and display packets

## How to pull out and push back values for fields



Need to quickly reference information to:

**Determine how  
traffic is structured**

**Where it is coming  
from**

**Ability to extract  
and push back is  
inside a larger  
Python program**



Scapy provides a finer level of control when performing tasks



```
###[ Ethernet ]###
dst      = 00:09:b0:1e:b7:cd
src      = 1c:1b:0d:0f:71:e4
type     = IPv4
```

```
###[ IP ]###
version  = 4
ihl      = 5
tos      = 0x0
len      = 157
id       = 54435
flags    = DF
frag     = 0
ttl      = 128
proto    = tcp
chksum   = 0xa19c
src      = 192.168.1.95
dst      = 192.168.1.107
\options \
```

```
###[ TCP ]###
sport    = 1836
dport    = 8009
seq      = 3115894920
ack      = 1069867455
dataofs  = 5
reserved = 0
flags    = PA
window   = 1020
chksum   = 0xc393
urgptr   = 0
options  = []
```

```
###[ Raw ]###
load     =
```



```
Packet=Ether(src="00:0c:29:0b:fc:15")/IP(src="192.168.1.147",  
dst="192.168.1.165", proto="tcp")/TCP(sport=12345, dport=22)
```



```

>>> ls(Packet)
dst          : DestMACField          = '00:50:56:b3:dc:8f' ('None')
src          : SourceMACField        = '00:0c:29:0b:fc:15' ('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         = 6                   ('0')
checksum    : XShortField           = None                ('None')
src         : SourceIPField         = '192.168.1.147'    ('None')
dst         : DestIPField           = '192.168.1.165'    ('None')
options     : PacketListField      = []                  ('[]')
--
sport       : ShortEnumField        = 12345               ('20')
dport       : ShortEnumField        = 22                  ('80')
seq         : IntField              = 0                   ('0')
ack         : IntField              = 0                   ('0')
dataofs     : BitField (4 bits)     = None                ('None')
reserved    : BitField (3 bits)     = 0                   ('0')
flags       : FlagsField           = <Flag 2 (S)>        ('<Flag 2 (S)>')
window     : ShortField            = 8192                ('8192')
checksum    : XShortField           = None                ('None')
urgptr      : ShortField            = 0                   ('0')
options     : TCPOptionsField      = []                  ('b''')

```



# Ethernet Frame Layout

Ethernet Frame (in 0x)

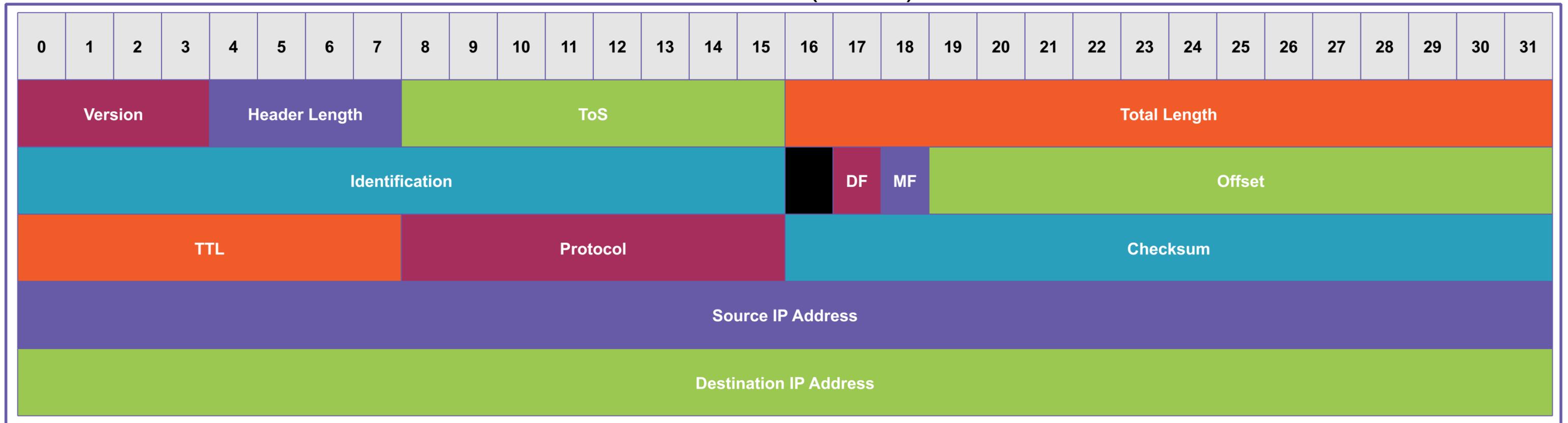


```
>>> ls(a[12])
dst      : DestMACField          = '00:f6:20:82:14:72' ('None')
src      : SourceMACField       = '1c:1b:0d:0f:71:e4' ('None')
type     : XShortEnumField      = 2048                ('36864')
...
```



# IP Packet Layout

IP Packet (in bits)



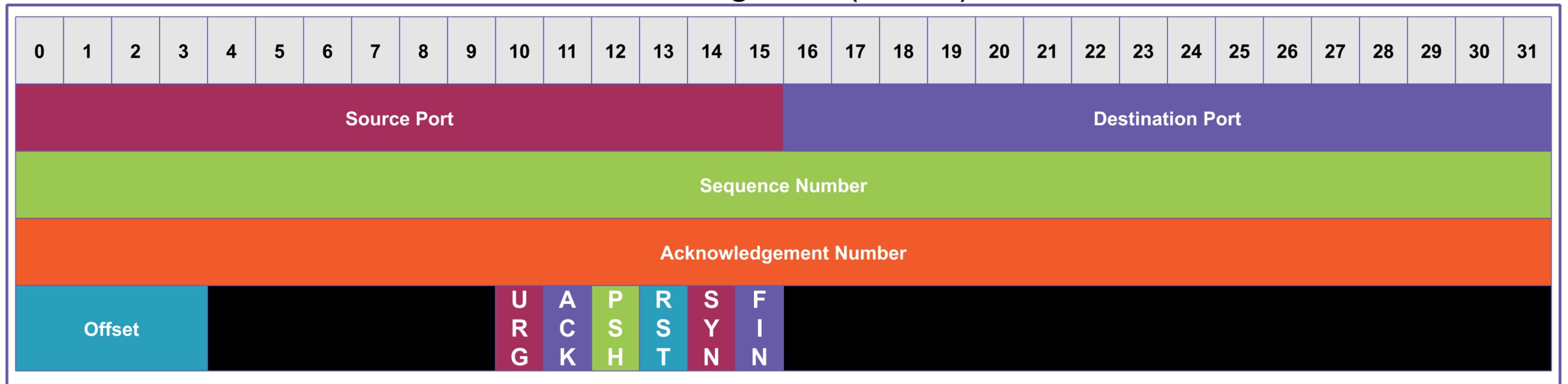
# ls command

```
>>> ls(a[12])
...
version      : BitField   (4 bits)      = 4          ('4')
ihl          : BitField   (4 bits)      = 5          ('None')
tos          : XByteField              = 0          ('0')
len          : ShortField              = 40         ('None')
id           : ShortField              = 65233      ('1')
flags        : FlagsField              = <Flag 2 (DF)> ('<Flag 0
()>')
frag         : BitField   (13 bits)     = 0          ('0')
ttl          : ByteField                = 128        ('64')
proto        : ByteEnumField           = 6          ('0')
checksum     : XShortField             = 30696      ('None')
src          : SourceIPField           = '192.168.1.95' ('None')
dst          : DestIPField             = '192.168.1.102' ('None')
options      : PacketListField         = []         ('[]')
...
```



# TCP Header

TCP Segment (in bits)



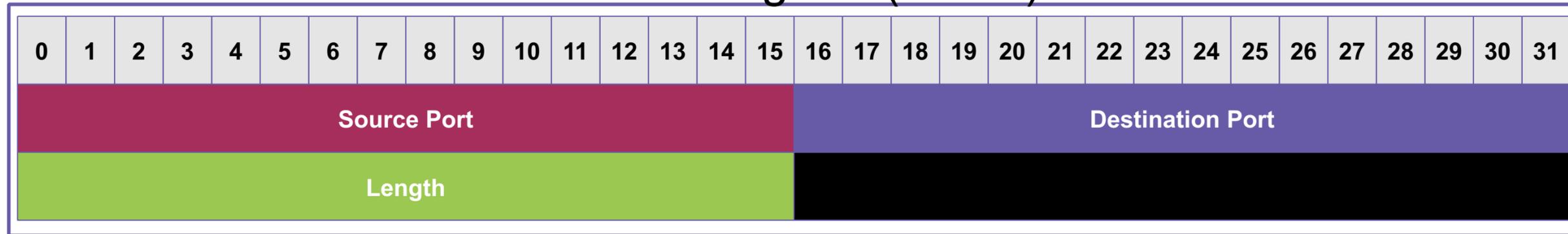
# ls command

```
>>> ls(a[12])
...
sport      : ShortEnumField      = 2724      ('20')
dport      : ShortEnumField      = 8009      ('80')
seq        : IntField            = 2903138928 ('0')
ack        : IntField            = 2600494214 ('0')
dataofs    : BitField (4 bits)   = 5         ('None')
reserved   : BitField (3 bits)   = 0         ('0')
flags      : FlagsField          = <Flag 16 (A)> ('<Flag 2
(S)>')
window     : ShortField          = 1022      ('8192')
chksum     : XShortField         = 61137     ('None')
urgptr     : ShortField          = 0         ('0')
options    : TCPOptionsField     = []        ("b' '")
--
load       : StrField            = b'\x00\x00\x00\x00\x00\x00'
("b' '")
```



# UDP Header

## UDP Datagram (in bits)



```
>>> ls(a[6])
```

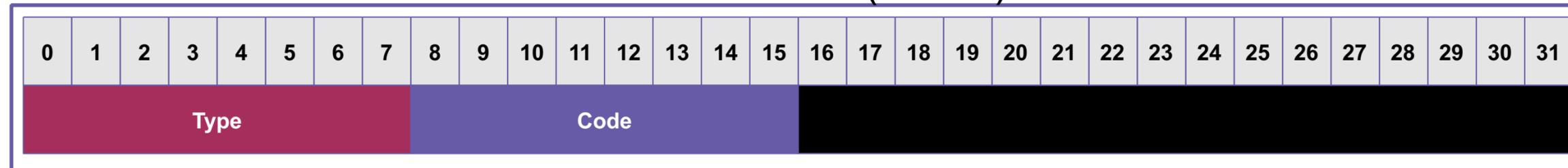
```
...
```

```
sport      : ShortEnumField      = 9999      ('53')
dport      : ShortEnumField      = 9999      ('53')
len        : ShortField          = 71        ('None')
chksum     : XShortField         = 4597      ('None')
--
load      : StrField             =
```



# ICMP

## ICMP Packet (in bits)



```
...
type      : ByteEnumField                = 8                ('8')
code      : MultiEnumField (Depends on 8) = 0                ('0')
chksum    : XShortField                  = 36411            ('None')
id        : XShortField (Cond)           = 27978            ('0')
seq       : XShortField (Cond)           = 4                ('0')
ts_ori    : ICMPTimeStampField (Cond)    = None             ('11512539')
ts_rx     : ICMPTimeStampField (Cond)    = None             ('11512539')
ts_tx     : ICMPTimeStampField (Cond)    = None             ('11512539')
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''')
--
load     : StrField                      =
```



# Scapy Range Support

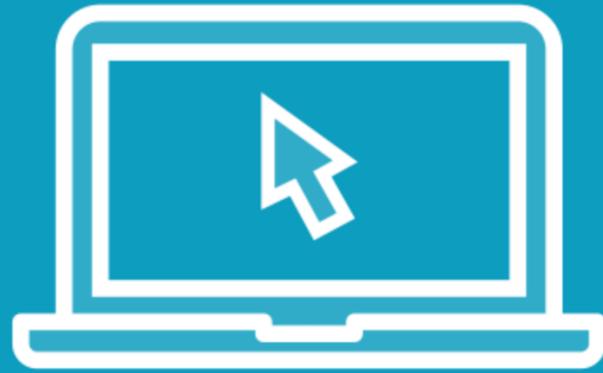
[1, 2, 3]

Support for ranges exists

```
z=Ether()/IP(dst=["192.168.1.0/29"])
```



# Demo



## Demonstrating Scapy Commands:

**.show**

**.summary**

**.hexdump**

**.filter**

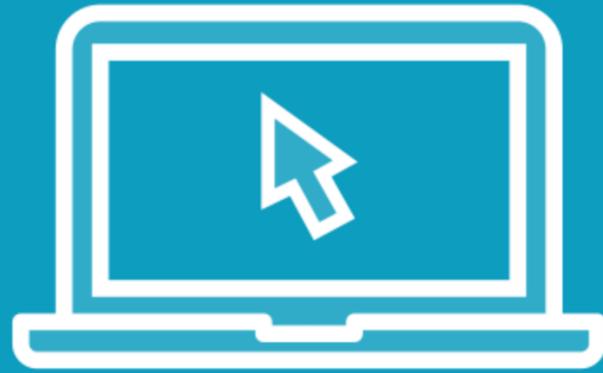
**ls**

**.show2**

**.sprintf**



Demo



**Basic IP Packet**

**Basic TCP Packet**

**Basic UDP Packet**

**Basic ICMP Packet**

**Displaying a Packet (ls command)**

**Pulling out Packet Fields**

**Altering Packet Fields**



# Summary



- **Reviewing Common Scapy Packet Display Commands and Methods**
- **Concepts Demonstration - Display Commands**
- **Methods of Creating and Dissecting Traffic**
- **Concepts Demonstration - Creating and Dissecting Traffic**

