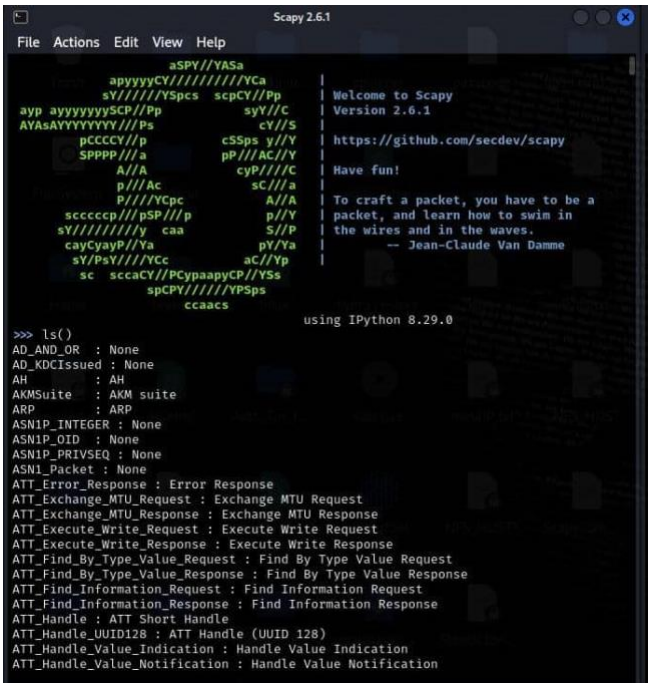
The image contains detailed information about Scapy, focusing on basics, packet crafting, layer fields, and altering packets. Let me transcribe and format the content into clean and presentable text.

Scapy Basics

To list supported layers: python

>>> ls()

Some key layers are: arp, ip, ipv6, tcp, udp, icmp.



To view layer fields, use: python

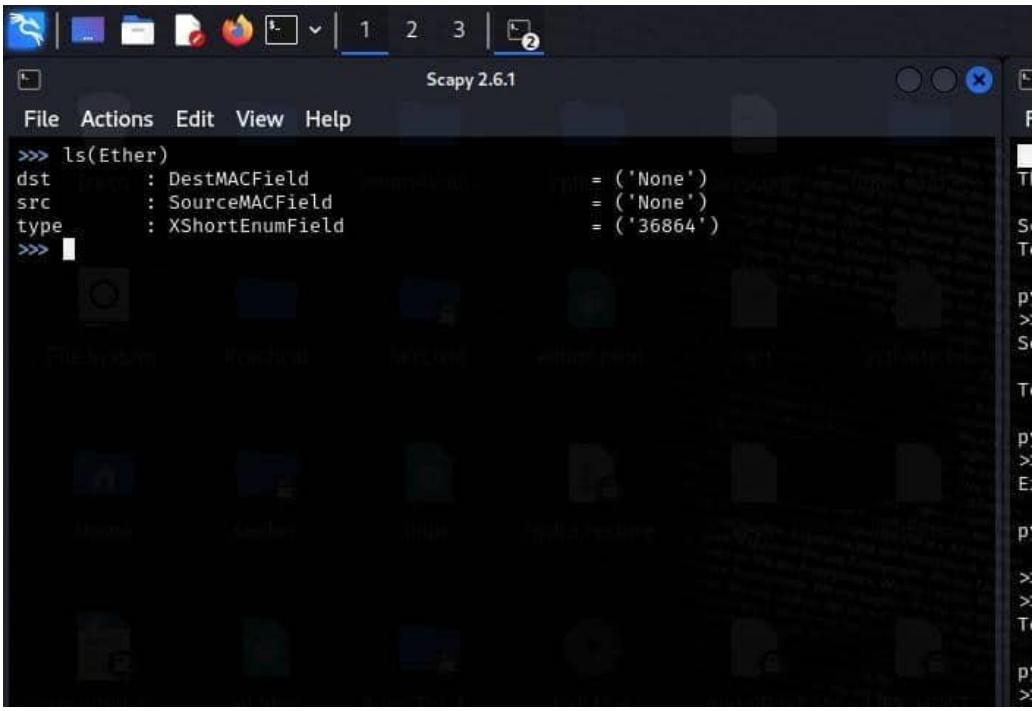
>>> ls(layer)

Example:

python

>>> ls(IPV6)

>>> ls(TCP)



To list the available commands: python

>>> lsc()

Key commands for interacting with packets:

rdpcap, send, sr, sniff, wrpcap.

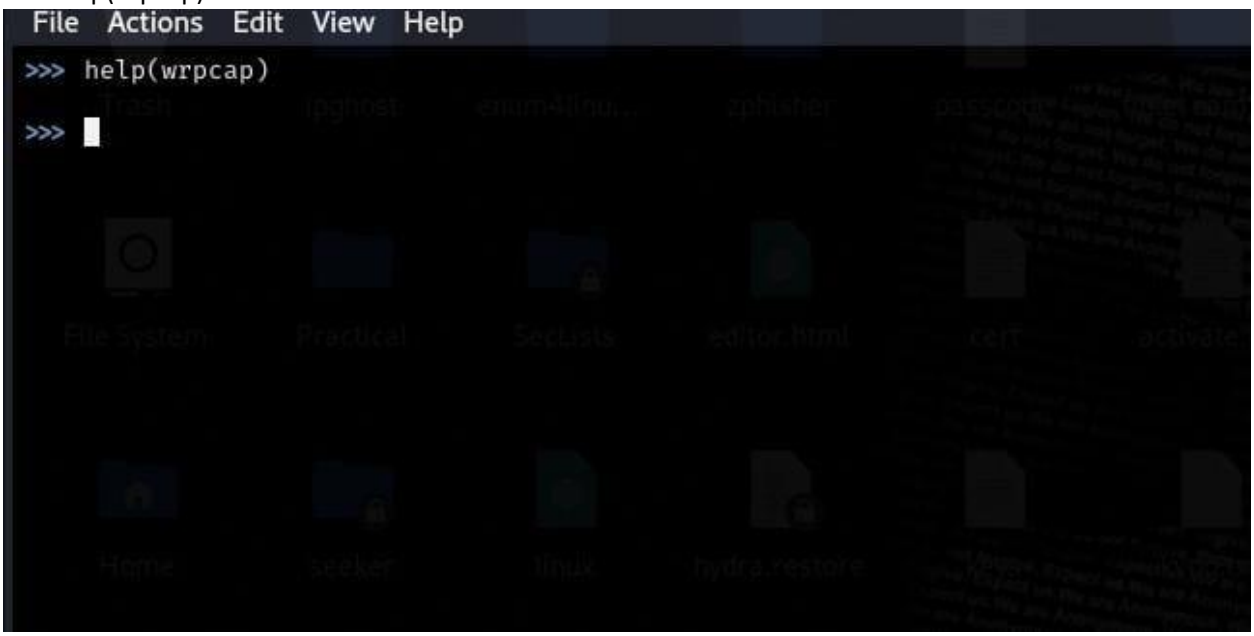
```
File  Actions  Edit  View  Help
>>> lsc()
IPID_count          : Identify IP id values classes in a list of packets
arp_mitm            : ARP MitM: poison 2 target's ARP cache
arpcachepoison      : Poison targets' ARP cache
arping              : Send ARP who-has requests to determine which hosts are up
::
arpleak             : Exploit ARP leak flaws, like NetBSD-SA2017-002.
bind_layers         : Bind 2 layers on some specific fields' values.
bridge_and_sniff    : Forward traffic between interfaces if1 and if2, sniff and
 return
chexdump            : Build a per byte hexadecimal representation
computeNIGroupAddr  : Compute the NI group Address. Can take a FQDN as input pa
rameter
connect_from_ip     : Open a TCP socket to a host:port while spoofing another I
P.
corrupt_bits        : Flip a given percentage (at least one bit) or number of b
its
corrupt_bytes       : Corrupt a given percentage (at least one byte) or number
of bytes
dclocator           : Perform a DC Locator as per [MS-ADTS] sect 6.3.6 or RFC41
20.
defrag              : defrag(plist) → ([not fragmented], [defragmented],
defragment          : defragment(plist) → plist defragmented as much as possib
le
dhcp_request        : Send a DHCP discover request and return the answer.
```

Get help with commands: python

>>> help(command)

Example: python

>>> help(rdpcap)

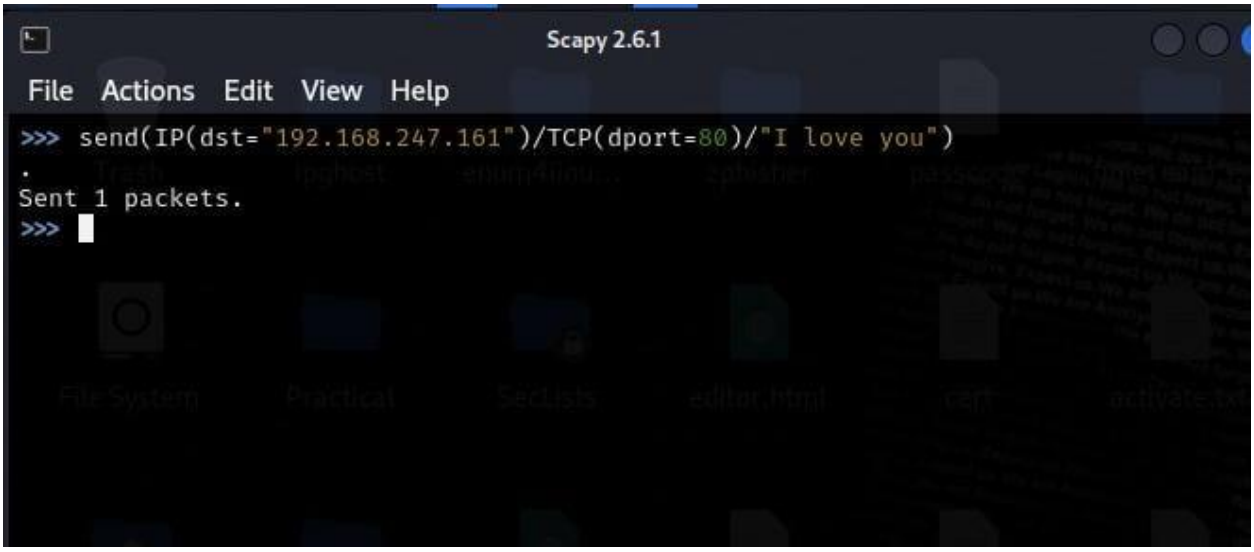


Basic Packet Crafting and Viewing

Scapy works with layers. Layers are functions linked together with the / character to construct packets.To build a basic TCP/IP packet with data as the payload: python

>>> send= IP(dst="1.2.3.4")/TCP(dport=22)/"data"

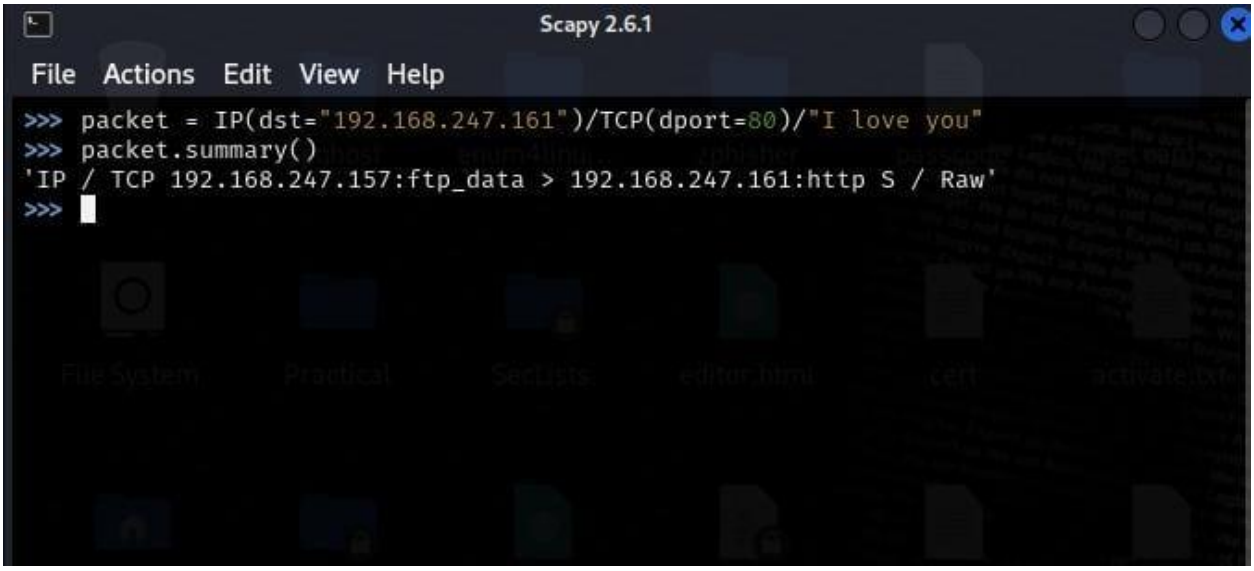
Note: Scapy allows crafting from the highest layer (e.g., TCP) to the lowest (Ethernet), using default field values if layers are omitted. Ensure proper order: Ethernet -> IP -> TCP.



To get a packet summary: python

>>> packet.summary()

To get more packet details:



python

>>> packet.show()

```
Scapy 2.6.1
File  Actions  Edit  View  Help
>>> packet = IP(dst="192.168.247.161")/TCP(dport=80)/"I love you"
>>> packet.summary()
'IP / TCP 192.168.247.157:ftp_data > 192.168.247.161:http S / Raw'
>>> packet.show()
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 192.168.247.157
  dst       = 192.168.247.161
  \options   \
###[ TCP ]###
     sport     = ftp_data
     dport     = http
     seq       = 0
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []
###[ Raw ]###
        load      = b'I love you'

>>>
```

Layer Fields and Default Values

Ethernet Layer Fields python

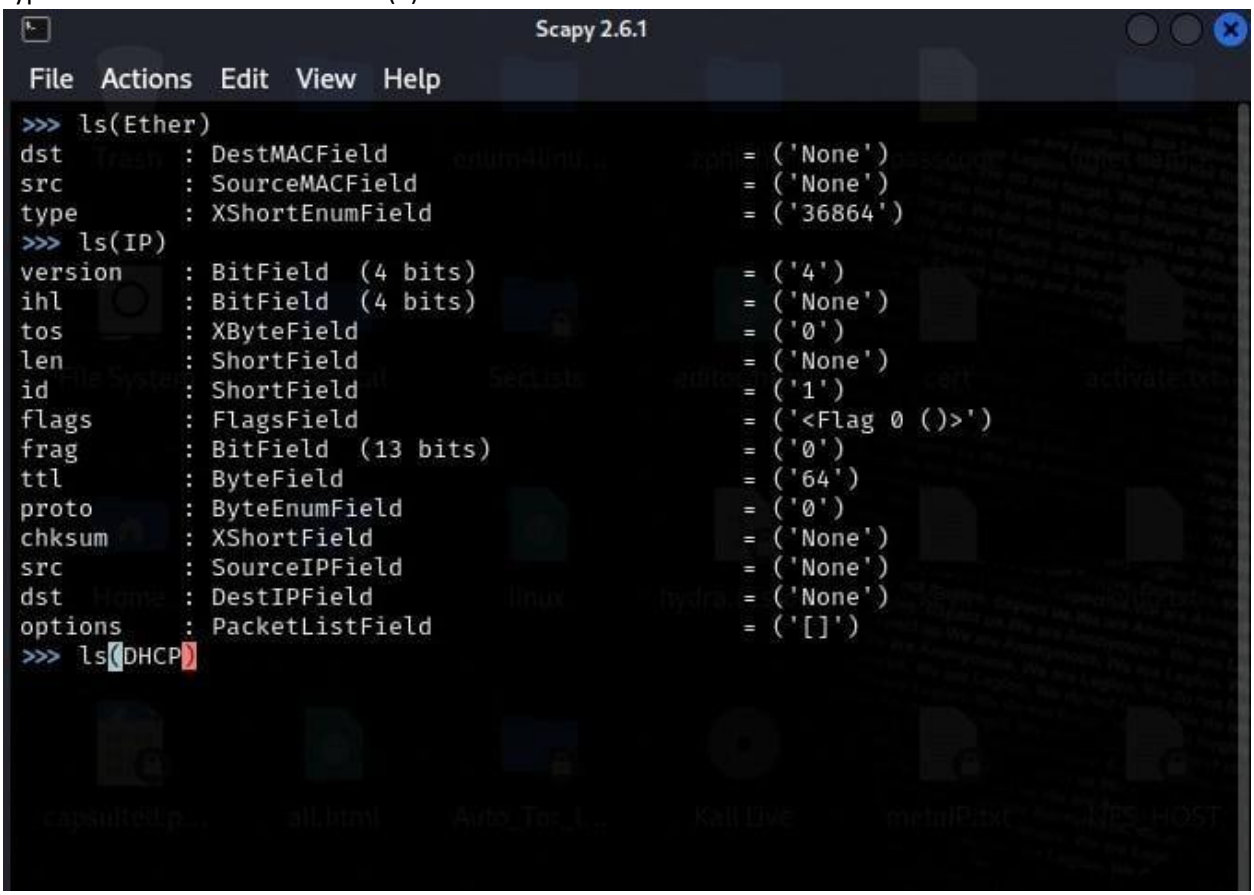
>>> ls(Ether)

Field Type Default Value

dst DestMACField (None)

src SourceMACField (None)

type XShortEnumField (0)



IPv4 Layer Fields: python

>>> ls(IP)

Field Type Default Value

version BitField (4)

ihl BitField (None)

tos XByteField (0)

len ShortField (None)

id ShortField (1)

flags FlagsField (0)

frag BitField (0)

ttl ByteField (64)

proto ByteEnumField (0)

chksum XShortField (None)

```
src        Emph    (None)
dst        Emph    (None)
options PacketListField   ([])
```

TCP Layer Fields

python

```
>>> ls(TCP)
Field      Type      Default Value
sport      ShortEnumField (20)
dport      ShortEnumField (80)
seq        IntField (0)
ack        IntField (0)
dataofs BitField (None)
reserved           BitField (0)
flags      FlagsField          (2)
windowShortField          (8192)
chksum XShortField        (None)
urgptr     ShortField          (0)
options TCPOptionsField             (())
```

Altering Packets

Packet layer fields are Python variables and can be modified.

Example Packet: python

>>> send = IP(dst="10.10.10.50")/TCP(sport=80)

```
Scapy 2.6.1
File  Actions  Edit  View  Help
>>> send=(IP(src="192.168.247.157" , dst="197.255.125.90")/TCP(sport=5666,dport=5
...: 900)/"/usr/share/metasploit-framework/modules/payloads/singles/linux/x64/she
...: ll_bind_tcp_random_port.rb")
>>> print(send.show())
###[ IP ]###
  version   = 4
  ihl       = None
  tos       = 0x0
  len       = None
  id        = 1
  flags     =
  frag      = 0
  ttl       = 64
  proto     = tcp
  chksum    = None
  src       = 192.168.247.157
  dst       = 197.255.125.90
  \options   \
###[ TCP ]###
     sport     = nrpe
     dport     = 5900
     seq       = 0
     ack       = 0
     dataofs   = None
     reserved  = 0
     flags     = S
     window    = 8192
     chksum    = None
     urgptr    = 0
     options   = []
###[ Raw ]###
        load      = b'/usr/share/metasploit-framework/modules/payloads/singles/li
nux/x64/shell_bind_tcp_random_port.rb'

None
>>> send.sport
5666
>>>
```

Viewing a Field's Value (e.g., sport):

python

>>> send.sport

Setting the Source Port:

```
None
>>> send.sport
5666
>>>
```

python

>>> send.sport = 5666

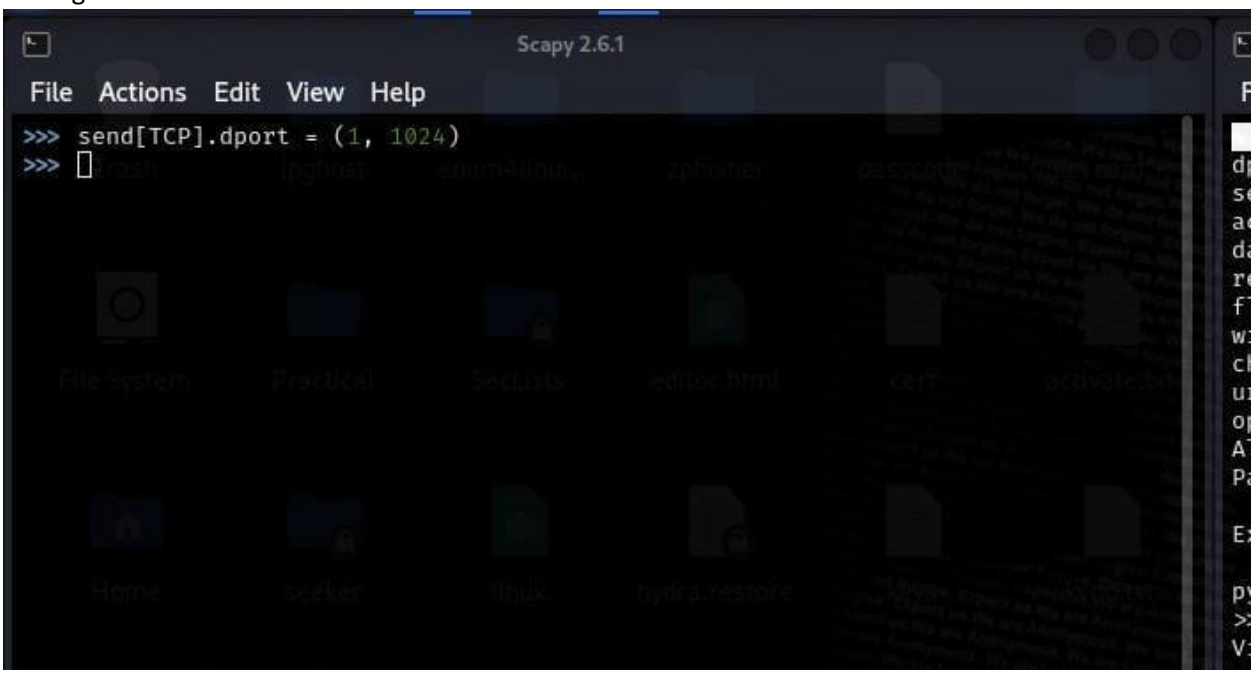
Setting Port Ranges:

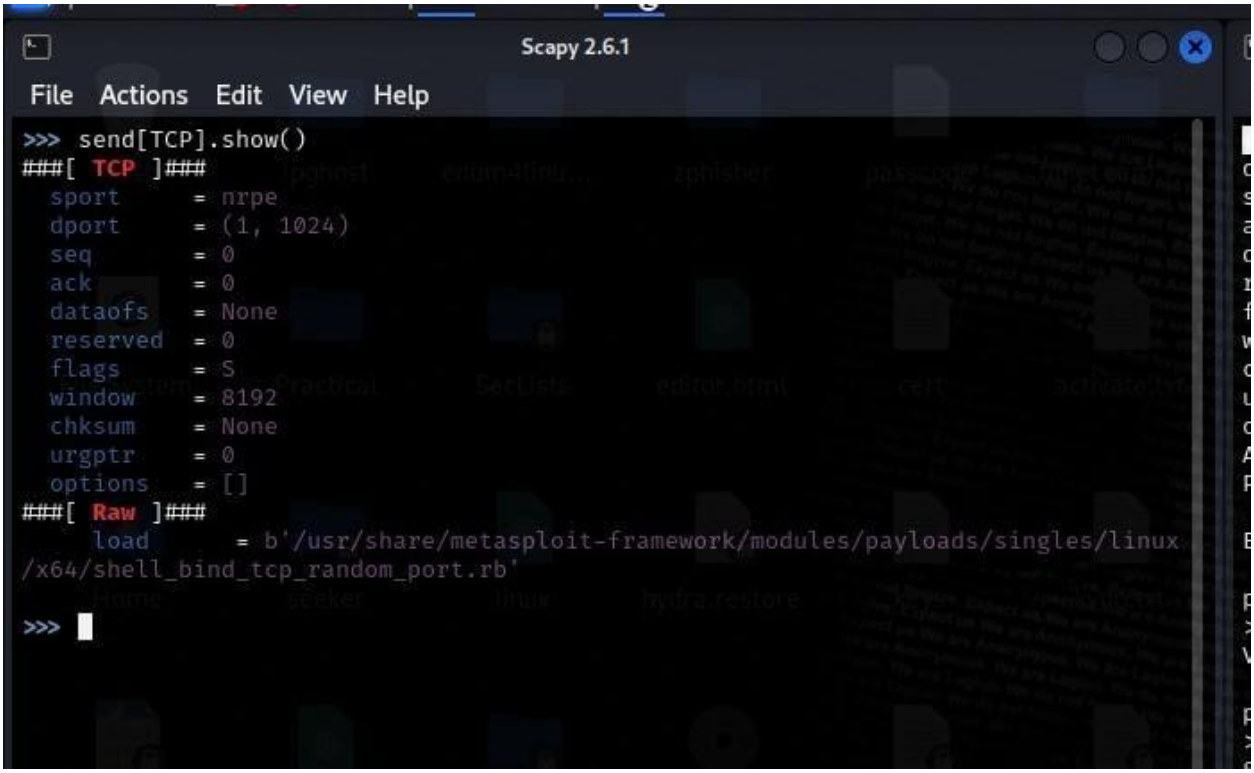
```
None
>>> send.sport
5666
>>>
```

Python

>>> send[TCP].dport = (1, 1024)

Setting a List of Ports:





python

>>> send[TCP].dport = [22, 80, 445]

Setting TCP Flags (Control Bits):

python

```
>>> send[TCP].flags = "SA"
>>> send[TCP].flags
18
>>> send.sprintf("%TCP.flags%")
'SA'
```

Setting Destination IP Address(es):python

```
>>> send[IP].dst = "1.2.3.4"
>>> send[IP].dst = ["1.2.3.4", "2.3.4.5", "5.6.7.8"]
```

Using CIDR:

python

```
>>> send[IP].dst = "1.2.3.4/16"
```

How to the output into a pcap or pcang file

>>>wrpcap(“capture.pcap” , send)