

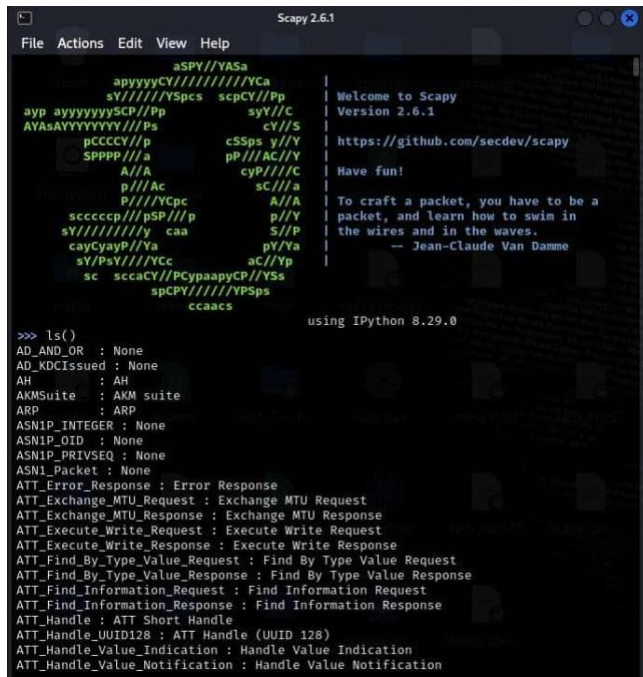
The image contains detailed information about Scapy, focusing on basics, packet crafting, layer fields, and altering packets. Let me transcribe and format the content into clean and presentable text.

Scapy Basics

To list supported layers: python

```
>>> ls()
```

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



```
Scapy 2.6.1
File Actions Edit View Help

      aSPY//YASa
  apyyyyCY/////////YCa
    sY/////////Yspcs  scpCY//Pp
ayp ayyyyyySCP//Pp    syY//C
AYAsAYYYYYYYY//Ps    cY//S
      pCCCCV//p      cSSps y//Y
      SPPPP//a        pP//AC//Y
      A//A            cyP///C
      p//Ac            sC///a
      P///YCpc        A//A
      scccccp///pSP///p    p//Y
      sY/////////y  caa    S//P
      cayCyayP//Ya    pY/Ya
      sY/PSY///YCC    ac//Yp
      sc  sccaCY//PCypaayCP//YSs
          apCPV/////////YPSps
          ccaacs

Welcome to Scapy
Version 2.6.1
https://github.com/secdev/scapy
Have fun!

To craft a packet, you have to be a
packet, and learn how to swim in
the wires and in the waves.
-- Jean-Claude Van Damme

using IPython 8.29.0

>>> ls()
AD_AND_OR : None
AD_KDCIssued : None
AH : AH
AKMSuite : AKM suite
ARP : ARP
ASN1P_INTEGER : None
ASN1P_OID : None
ASN1P_PRIVSEQ : None
ASN1_Packet : None
ATT_Error_Response : Error Response
ATT_Exchange_MTU_Request : Exchange MTU Request
ATT_Exchange_MTU_Response : Exchange MTU Response
ATT_Execute_Write_Request : Execute Write Request
ATT_Execute_Write_Response : Execute Write Response
ATT_Find_By_Type_Value_Request : Find By Type Value Request
ATT_Find_By_Type_Value_Response : Find By Type Value Response
ATT_Find_Information_Request : Find Information Request
ATT_Find_Information_Response : Find Information Response
ATT_Handle : ATT Short Handle
ATT_Handle_UUID128 : ATT Handle (UUID 128)
ATT_Handle_Value_Indication : Handle Value Indication
ATT_Handle_Value_Notification : Handle Value Notification
```

To view layer fields, use: python

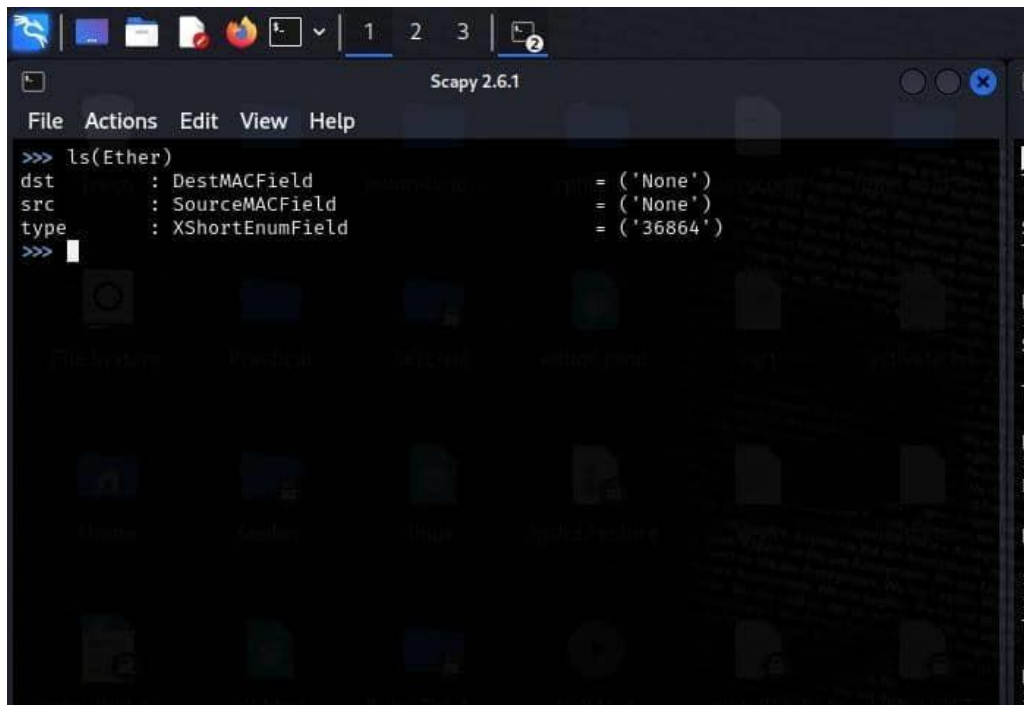
```
>>> ls(layer)
```

Example:

```
python
```

```
>>> ls(IPV6)
```

```
>>> ls(TCP)
```



To list the available commands: python

```
>>> lsc()
```

Key commands for interacting with packets:

rdpcap, send, sr, sniff, wrpcap.

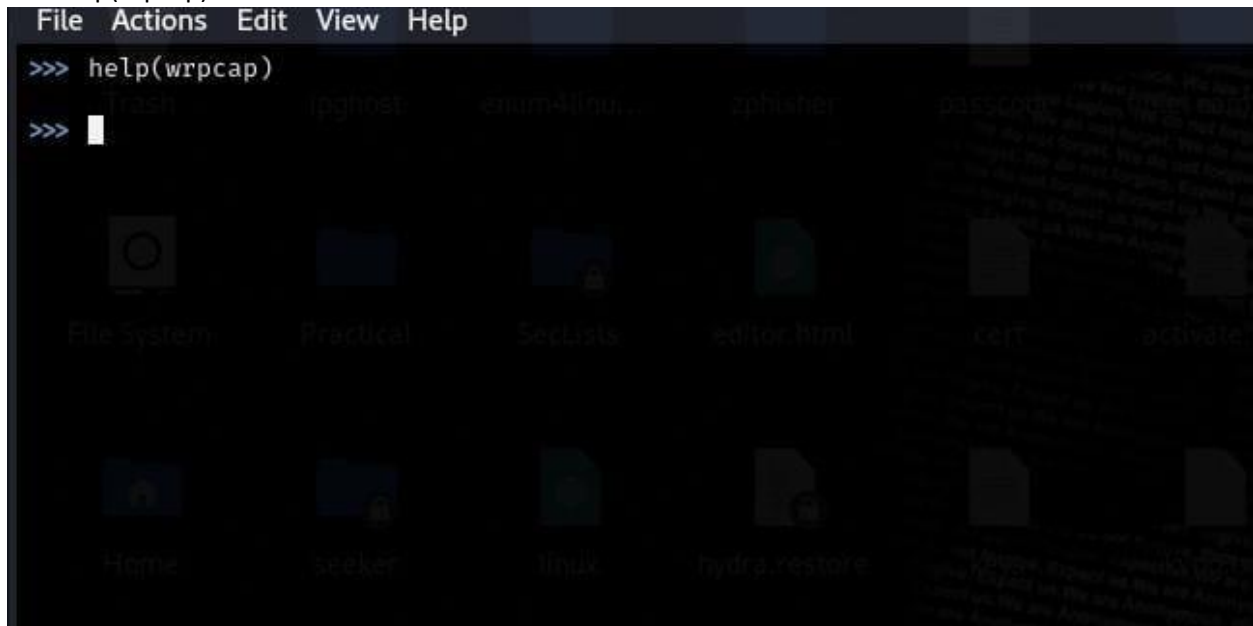
```
File Actions Edit View Help
>>> lsc()
IPID_count      : Identify IP id values classes in a list of packets
arp_mitm        : ARP MitM: poison 2 target's ARP cache
arpcachepoison  : Poison targets' ARP cache
arping         : Send ARP who-has requests to determine which hosts are up
::
arpleak        : 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(rdpicap)
```

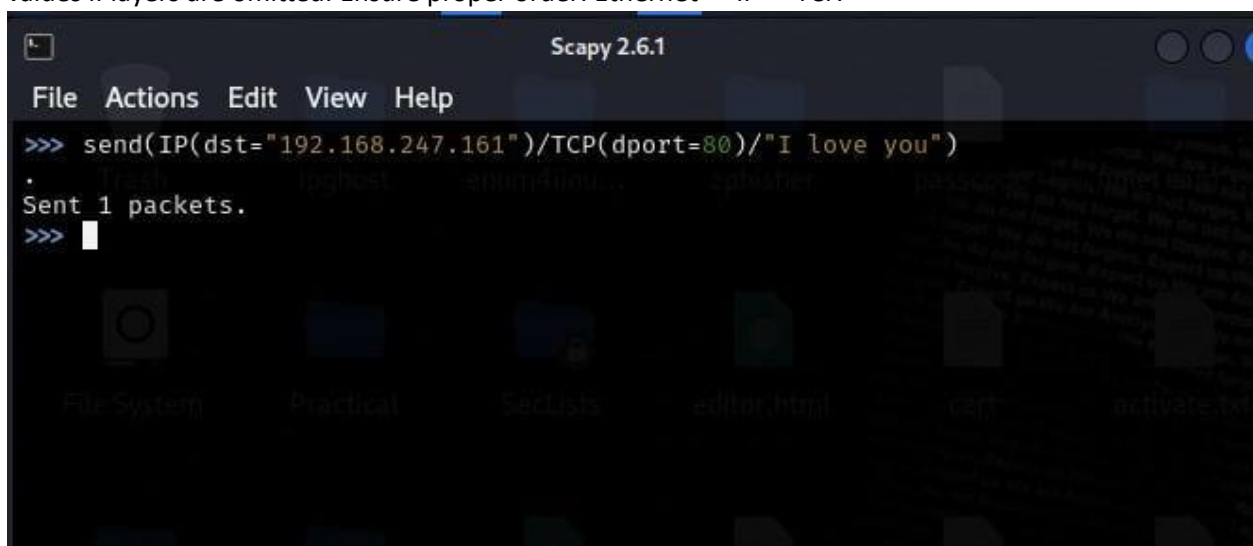


Basic Packet Crafting and Viewing

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")
```

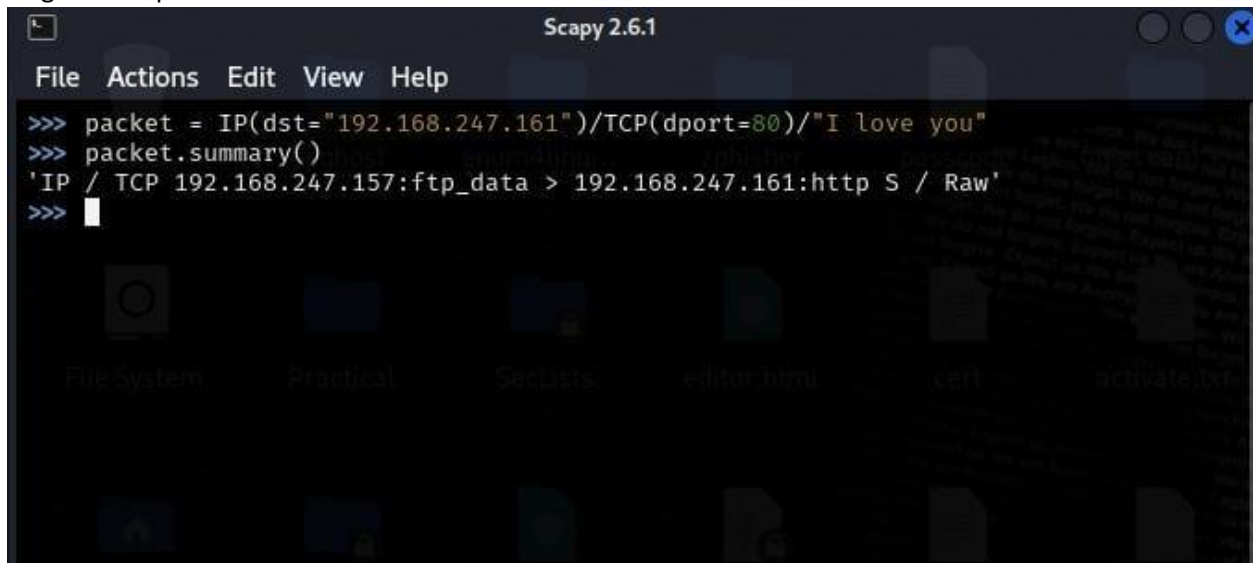
Note: Scapy allows crafting from the highest layer (e.g., TCP) to the lowest (Ethernet), using default field values if layers are omitted. Ensure proper order: Ethernet -> IP -> TCP.



To get a packet summary: python

```
>>> packet.summary()
```

To get more packet details:

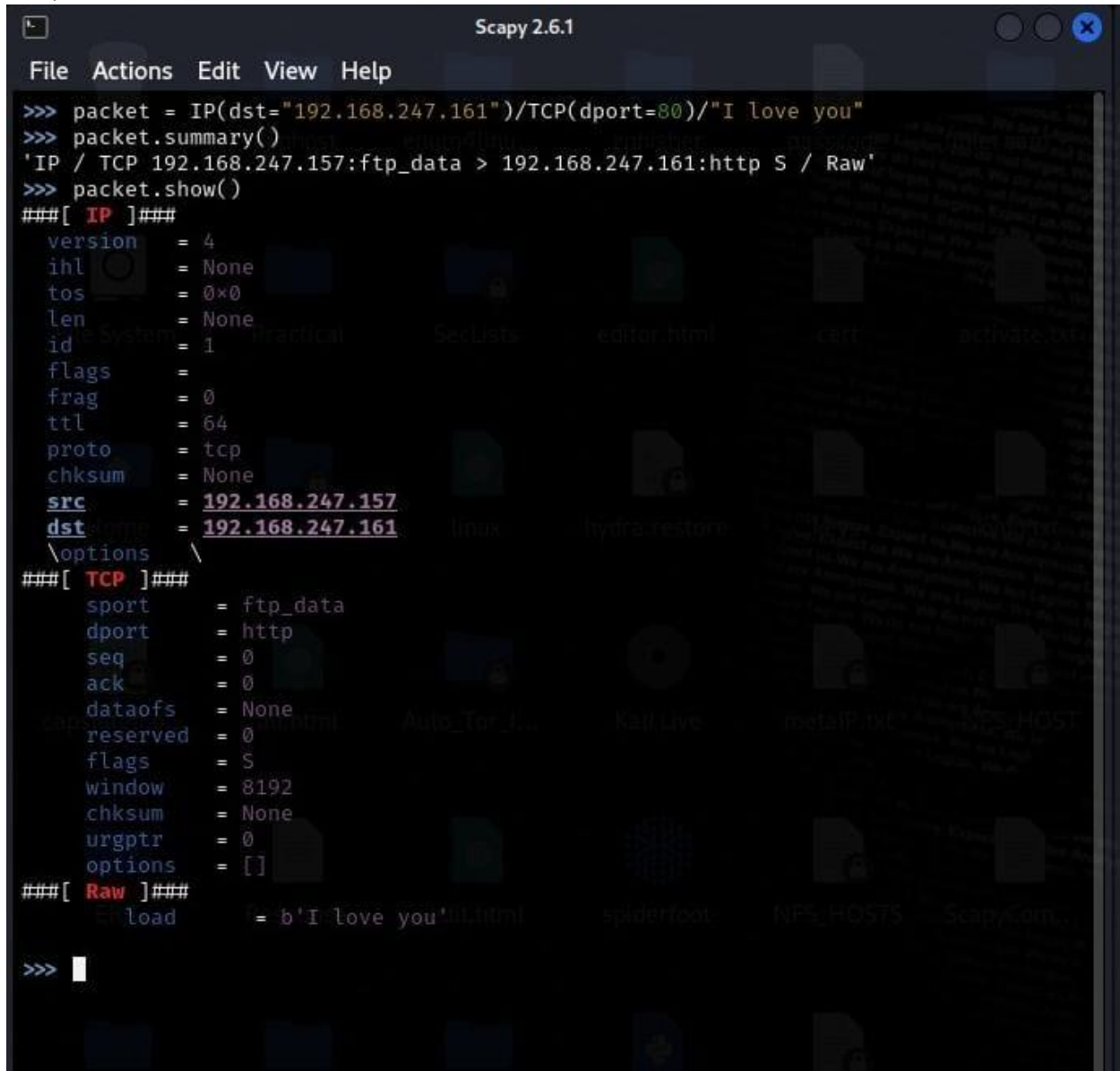
A screenshot of a terminal window titled "Scapy 2.6.1". The window has a menu bar with "File", "Actions", "Edit", "View", and "Help". The terminal shows the following commands and output:

```
>>> 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'
>>>
```

The terminal background is dark with a faint, repeating pattern of text. Below the terminal window, there are several icons representing different file systems or folders, including "File System", "Practical", "SecLists", "editor.html", "cert", and "activate.txt".

python

```
>>> packet.show()
```



The screenshot shows a Scapy 2.6.1 terminal window with a dark background. The menu bar includes 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal output shows the following commands and results:

```
>>> 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'
>>>
```

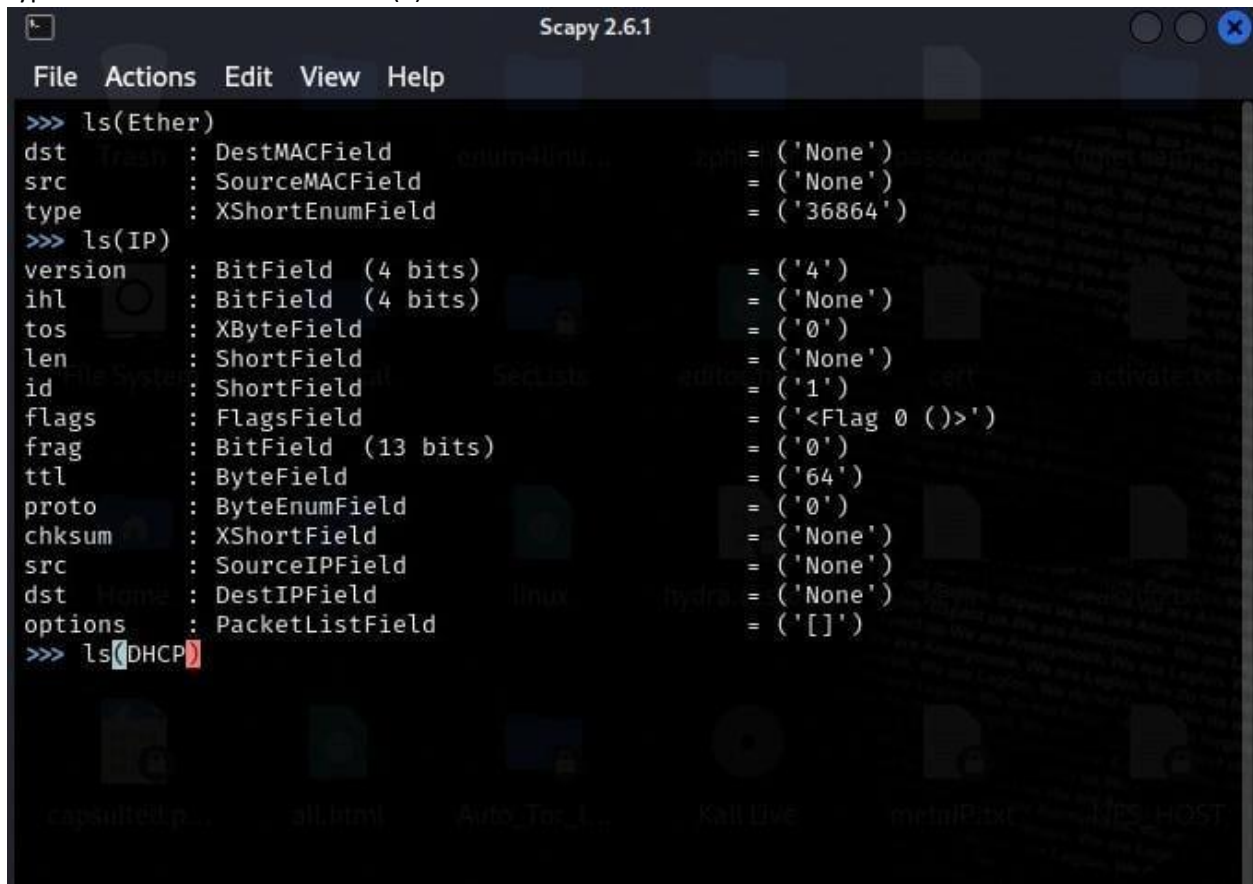
Layer Fields and Default Values

Ethernet Layer Fields python

```
>>> ls(Ether)
```

Field	Type	Default Value
dst	DestMACField	(None)
src	SourceMACField	(None)

type XShortEnumField (0)

A screenshot of the Scapy 2.6.1 application window. The window has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The main area is a terminal-like interface with a dark background and light-colored text. It shows the following commands and their outputs:

```
>>> ls(Ether)
dst      : DestMACField      = ('None')
src      : SourceMACField    = ('None')
type     : XShortEnumField   = ('36864')
>>> ls(IP)
version  : BitField (4 bits) = ('4')
ihl      : BitField (4 bits) = ('None')
tos      : XByteField        = ('0')
len      : ShortField        = ('None')
id       : ShortField        = ('1')
flags    : FlagsField       = ('<Flag 0 (>')
frag     : BitField (13 bits) = ('0')
ttl      : ByteField         = ('64')
proto    : ByteEnumField     = ('0')
chksum   : XShortField       = ('None')
src      : SourceIPField     = ('None')
dst      : DestIPField       = ('None')
options  : PacketListField   = ('[]')
>>> ls(DHCP)
```

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

>>> Is(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)
-------	------------	-----

window	ShortField	(8192)
--------	------------	--------

chksum	XShortField	(None)
--------	-------------	--------

urgptr	ShortField	(0)
--------	------------	-----

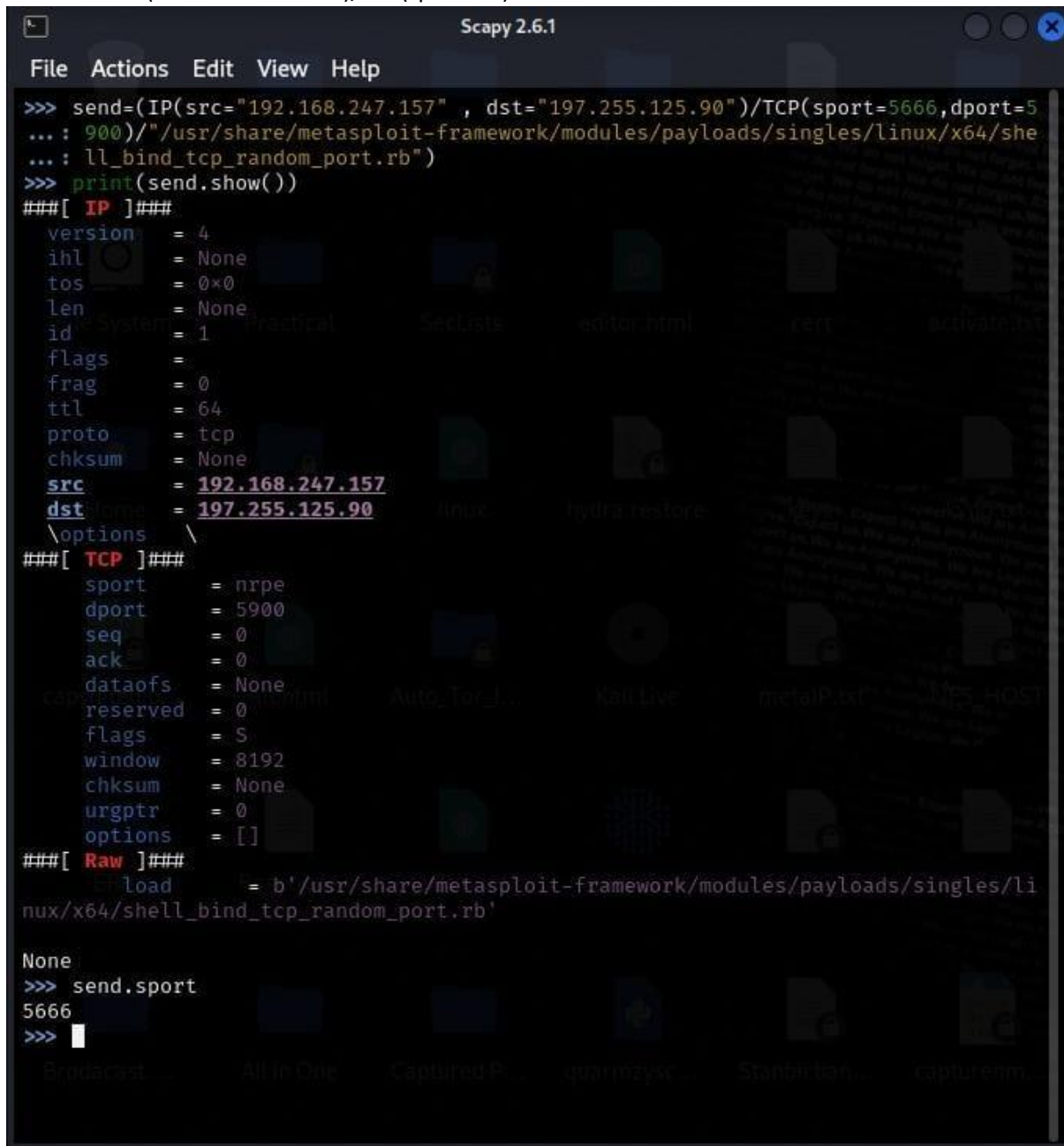
options	TCPOptionsField	(())
---------	-----------------	------

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)
```

A screenshot of a Scapy 2.6.1 terminal window. The window has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal shows the following commands and output:

```
>>> 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
```

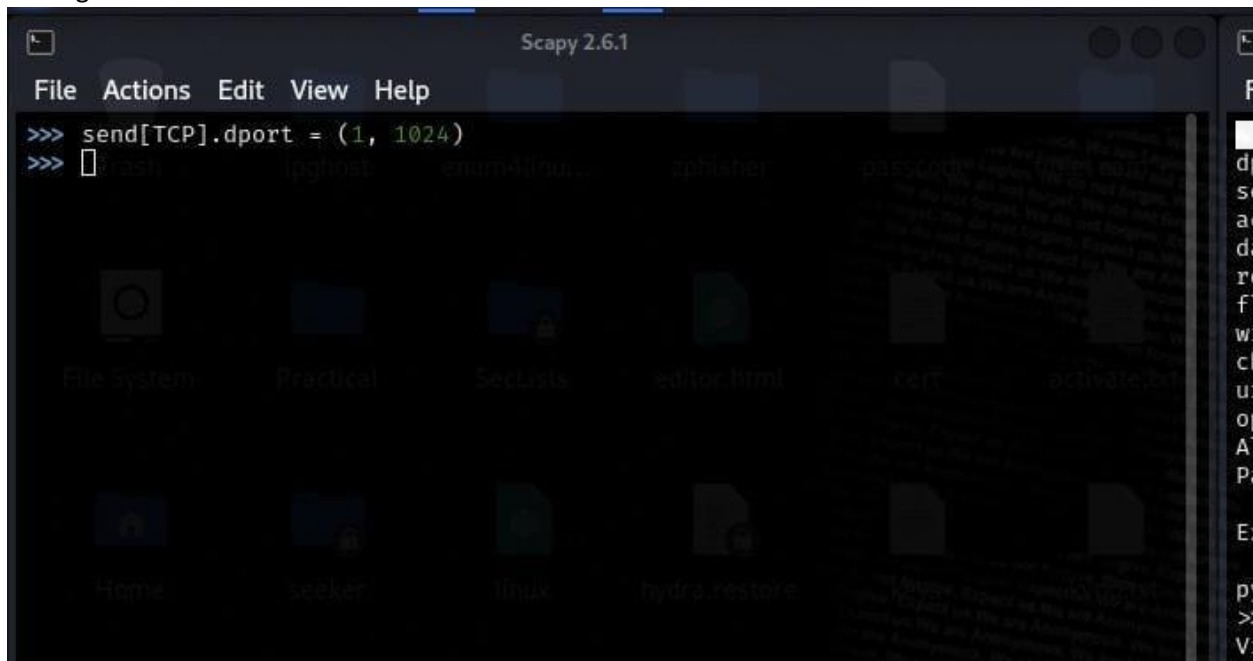
Setting Port Ranges:

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

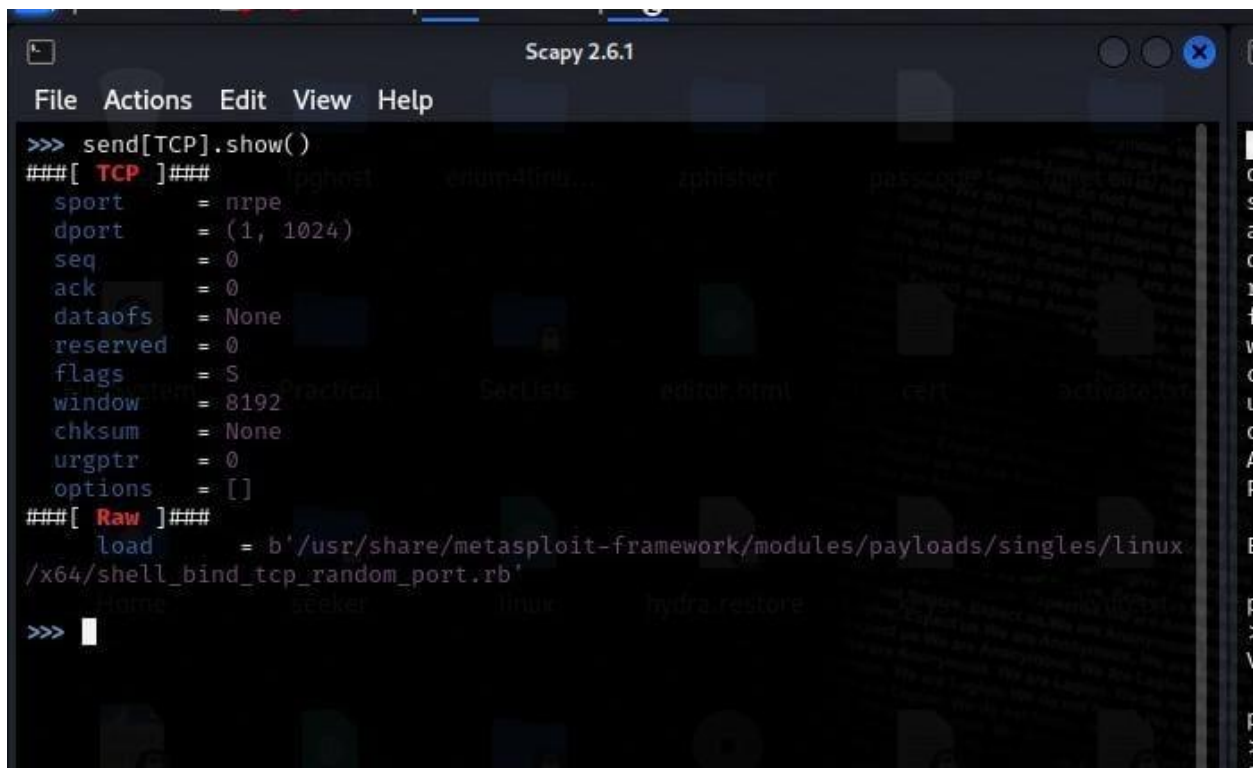
Python

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

Setting a List of Ports:



```
Scapy 2.6.1
File Actions Edit View Help
>>> send[TCP].dport = (1, 1024)
>>> 
```



```
Scapy 2.6.1
File Actions Edit View Help
>>> send[TCP].show()
###[ TCP ]###
  sport    = nrpe
  dport    = (1, 1024)
  seq      = 0
  ack      = 0
  dataofs  = None
  reserved = 0
  flags    = S
  window   = 8192
  checksum = None
  urgptr   = 0
  options  = []
###[ Raw ]###
  load     = b'/usr/share/metasploit-framework/modules/payloads/singles/linux/x64/shell_bind_tcp_random_port.rb'
>>> 
```

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"]
```

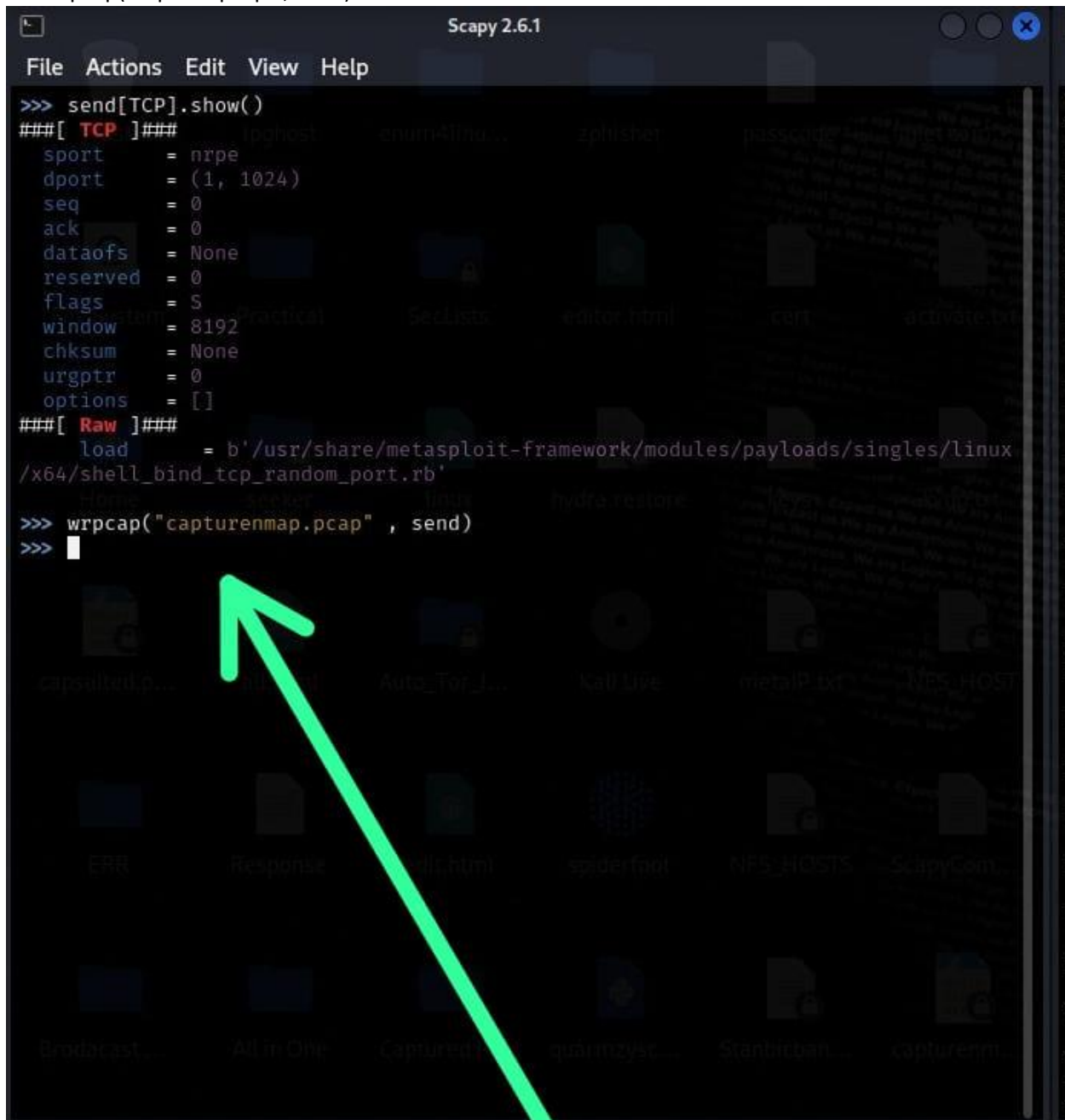
Using CIDR:

python

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

How to the output into a pcap or pcang file

```
>>> wrpcap("capture.pcap", send)
```



The screenshot shows a terminal window titled "Scapy 2.6.1" with a menu bar (File, Actions, Edit, View, Help). The terminal output displays the details of a TCP packet captured on interface 'nrpe' at port 1024. The packet has sequence number 0, acknowledgment number 0, and window size 8192. The raw data is shown as a hexadecimal string. The user then enters the command `wrpcap("capture.pcap", send)` to save the packet to a file. A red arrow points to the filename "capture.pcap" in the command.

```
>>> send[TCP].show()
###[ TCP ]###
sport      = nrpe
dport      = (1, 1024)
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/linux
/x64/shell_bind_tcp_random_port.rb'
>>> wrpcap("capture.pcap", send)
>>>
```