3.4 Scapy基础

Scapy是一个强大的交互式数据包处理程序（使用python编写）。它能够伪造或者解码大量的网络协议数据包，能够发送、捕捉、匹配请求和回复包等等。它可以很容易地处理一些典型操作，比如端口扫描，tracerouting，探测，单元 测试，攻击或网络发现（可替代hping，NMAP，arpspoof，ARP-SK，arping，tcpdump，tethereal，P0F等）。最重要的他还有很多更优秀的特性——发送无效数据帧、注入修改的802.11数据帧、在WEP上解码加密通道（VOIP）、ARP缓存攻击（VLAN）等，这也是其他工具无法处理完成的。

本章的大部分工具开发都是基于Scapy来开发的，所以各位同学务必熟悉Scapy的基本使用方法。

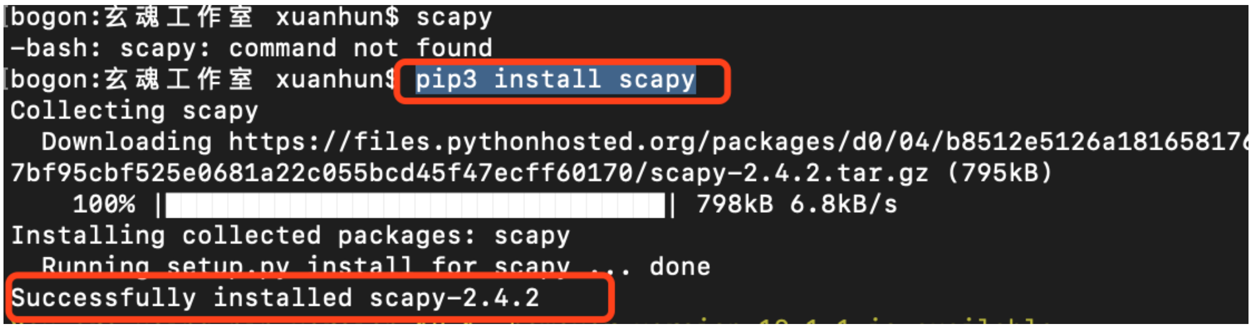
Scapy可以通过命令行和Python调用两种方式来进行使用。在使用之前请确保已经安装Scapy。

3.4.1 Scapy安装

通过命令

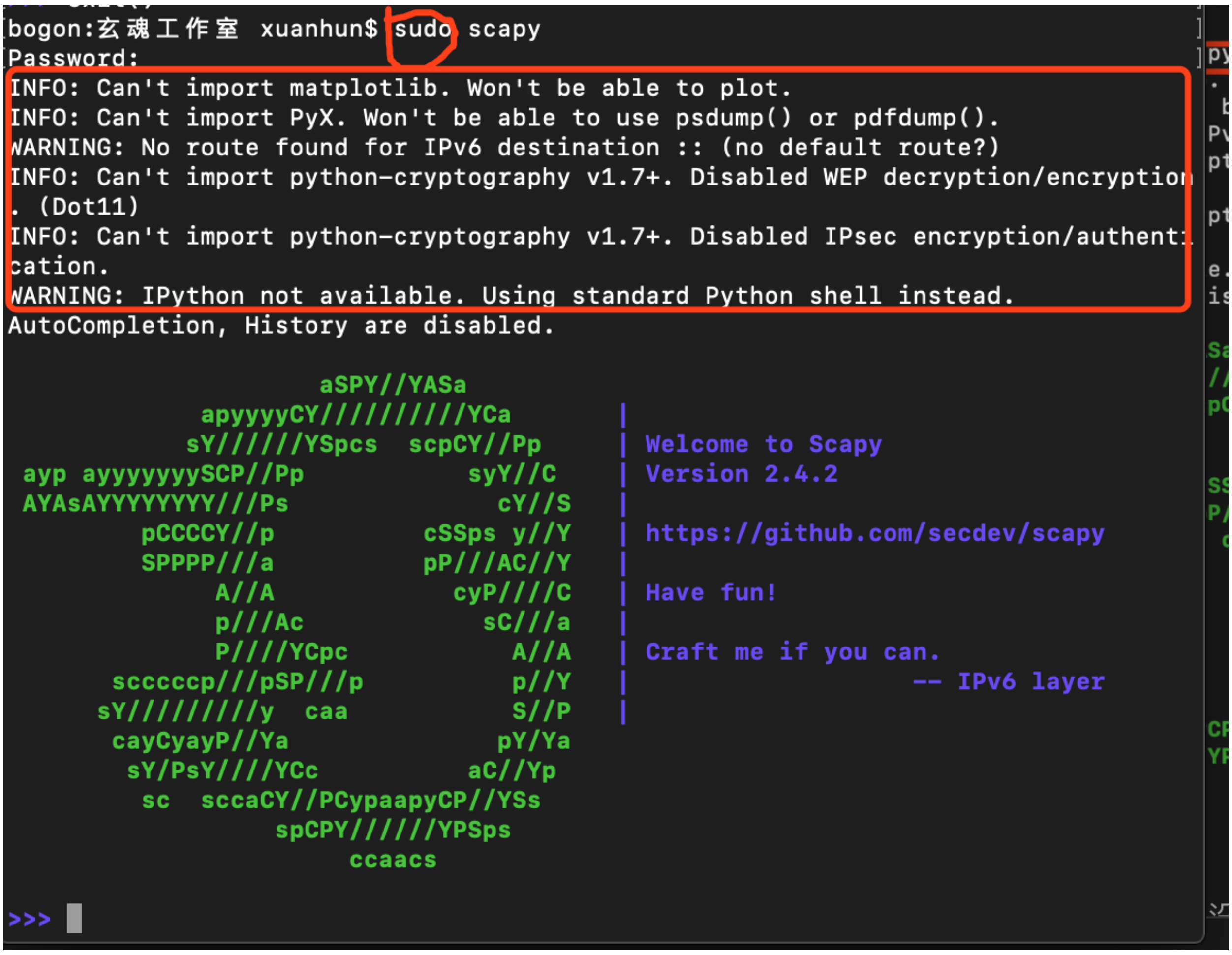
```
pip3 install scapy
```

来安装scapy。



```
bogon:玄魂工作室 xuanhun$ scapy
-bash: scapy: command not found
bogon:玄魂工作室 xuanhun$ pip3 install scapy
Collecting scapy
  Downloading https://files.pythonhosted.org/packages/d0/04/b8512e5126a18165817
7bf95cbf525e0681a22c055bcd45f47ecff60170/scapy-2.4.2.tar.gz (795kB)
    100% |████████████████████████████████| 798kB 6.8kB/s
Installing collected packages: scapy
  Running setup.py install for scapy ... done
Successfully installed scapy-2.4.2
```

安装之后可以在终端启动。因为发送数据包需要root权限，所以使用sudo启动。

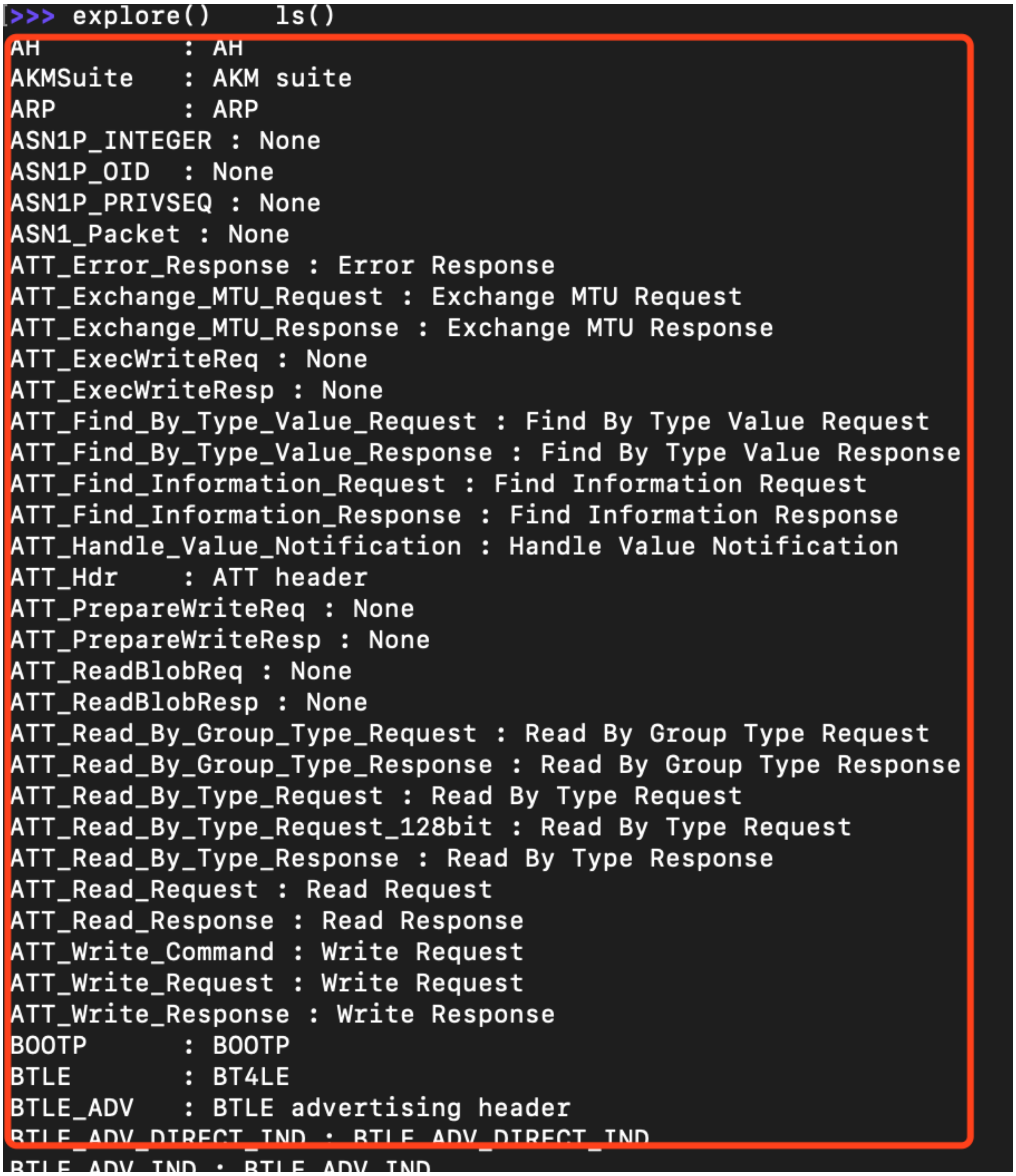


```
bogon:玄魂工作室 xuanhun$ sudo scapy
Password:
INFO: Can't import matplotlib. Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)
INFO: Can't import python-cryptography v1.7+. Disabled WEP decryption/encryption
. (Dot11)
INFO: Can't import python-cryptography v1.7+. Disabled IPsec encryption/authenti
cation.
WARNING: IPython not available. Using standard Python shell instead.
AutoCompletion, History are disabled.

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.2
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Craft me if you can.
       scccccp///pSP///p          p//Y   |                   -- IPv6 layer
      sY/////////y  caa           S//P   |
       cayCyayP//Ya              pY/Ya
        sY/PsY////YCc          aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs

>>> 
```

注意上图中的INFO信息，如果没有安装可选包，部分功能不可用，在需要的时候单独安装即可。

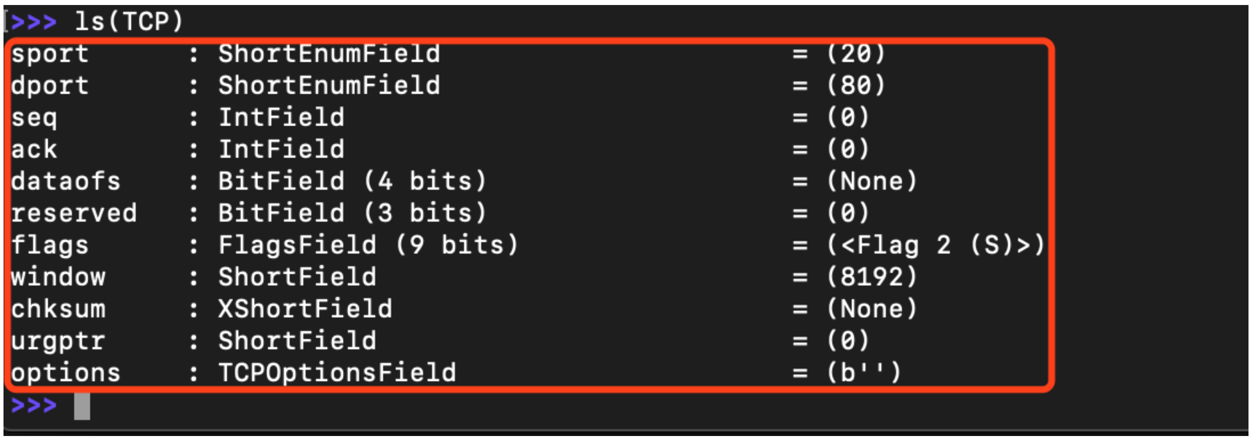
3.4.2 基本命令

ls()显示scapy支持的所有协议。



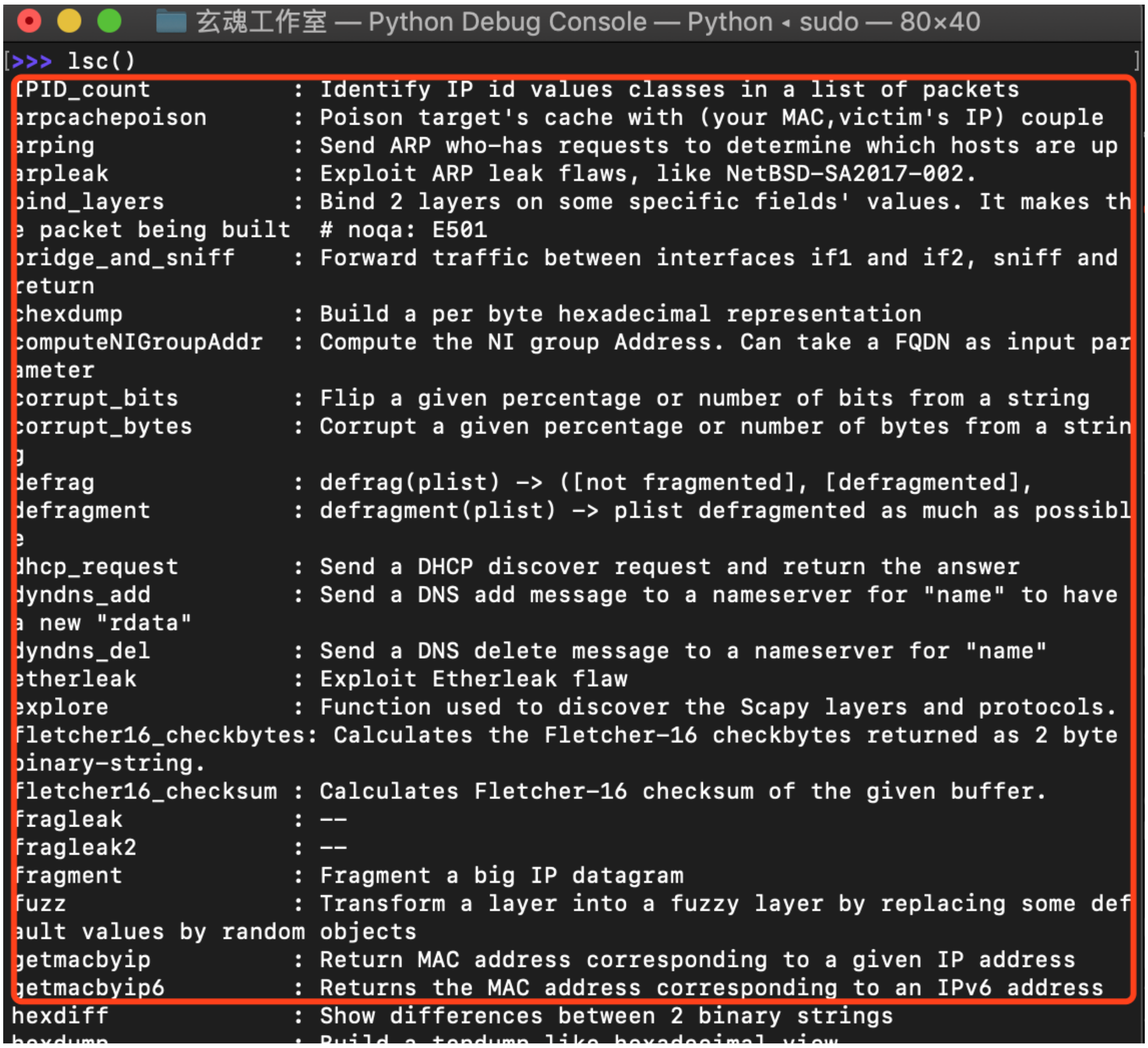
```
>>> explore()    ls()
AH           : AH
AKMSuite     : AKM suite
ARP          : ARP
ASN1P_INTEGER : None
ASN1P_OID    : None
ASN1P_PRIVSEQ : None
ASN1_Packet : None
ATT_Error_Response : Error Response
ATT_Exchange_MTU_Request : Exchange MTU Request
ATT_Exchange_MTU_Response : Exchange MTU Response
ATT_ExecWriteReq : None
ATT_ExecWriteResp : None
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_Value_Notification : Handle Value Notification
ATT_Hdr      : ATT header
ATT_PrepareWriteReq : None
ATT_PrepareWriteResp : None
ATT_ReadBlobReq : None
ATT_ReadBlobResp : None
ATT_Read_By_Group_Type_Request : Read By Group Type Request
ATT_Read_By_Group_Type_Response : Read By Group Type Response
ATT_Read_By_Type_Request : Read By Type Request
ATT_Read_By_Type_Request_128bit : Read By Type Request
ATT_Read_By_Type_Response : Read By Type Response
ATT_Read_Request : Read Request
ATT_Read_Response : Read Response
ATT_Write_Command : Write Request
ATT_Write_Request : Write Request
ATT_Write_Response : Write Response
BOOTP        : BOOTP
BTLE         : BT4LE
BTLE_ADV     : BTLE advertising header
BTLE_ADV_DIRECT_IND : BTLE_ADV_DIRECT_IND
BTLE_ADV_IND : BTLE_ADV_IND
```

这个命令足以体现Scapy的强大，上百种网络协议，直接秒杀其他工具。ls()函数的参数还可以是上面支持的协议中的任意一个的类型属性，也可以是任何一个具体的数据包，如ls(TCP),ls(newpacket)等。输入ls(TCP)会显示TCP方法构造对象的内容属性。



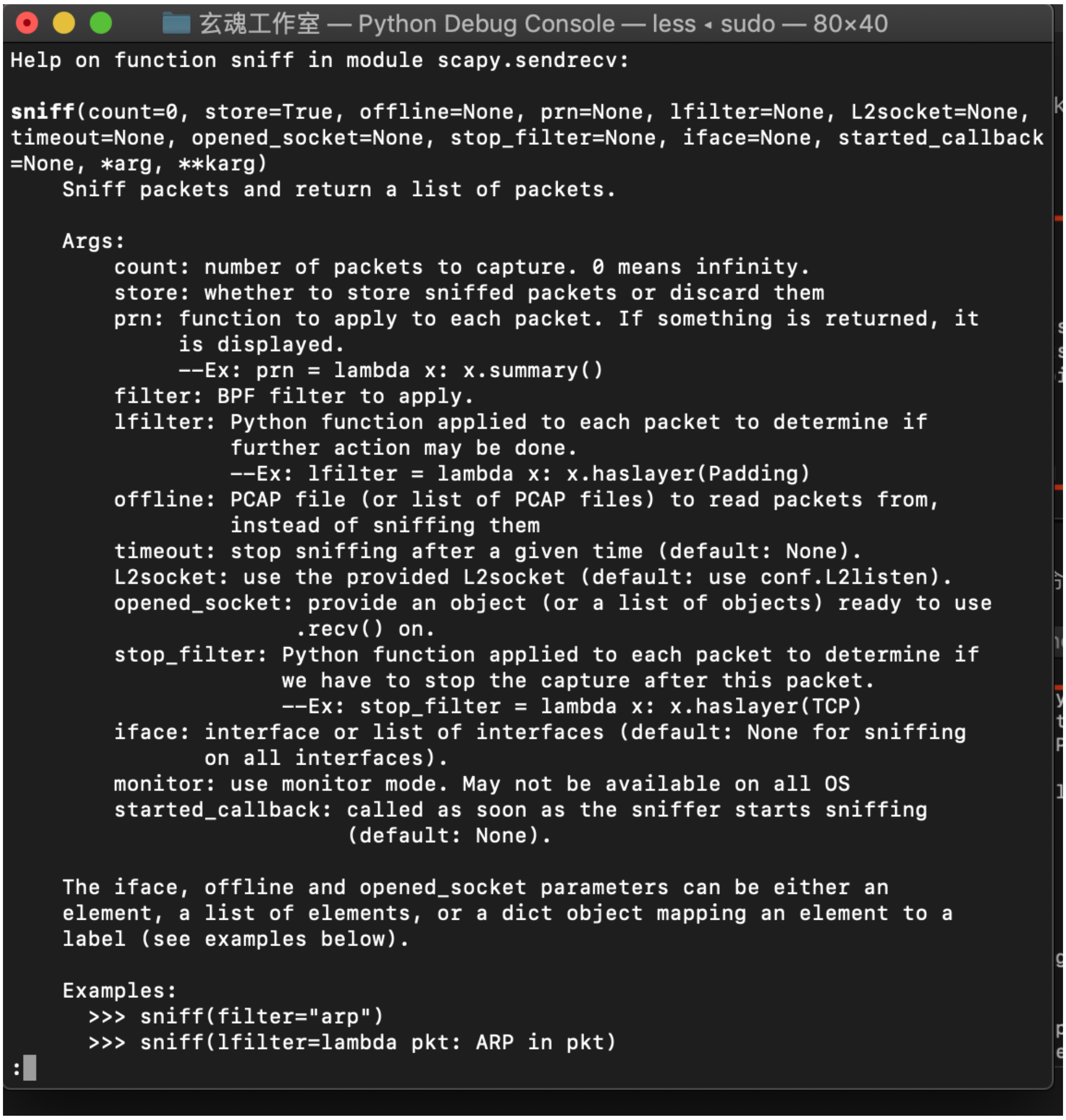
```
>>> ls(TCP)
sport      : ShortEnumField                        = (20)
dport      : ShortEnumField                        = (80)
seq        : IntField                              = (0)
ack        : IntField                              = (0)
dataofs    : BitField (4 bits)                     = (None)
reserved   : BitField (3 bits)                     = (0)
flags      : FlagsField (9 bits)                   = (<Flag 2 (S)>)
window     : ShortField                            = (8192)
chksum     : XShortField                           = (None)
urgptr     : ShortField                            = (0)
options    : TCPOptionsField                       = (b'')
>>> 
```

lsc()列出scapy支持的所有的命令。



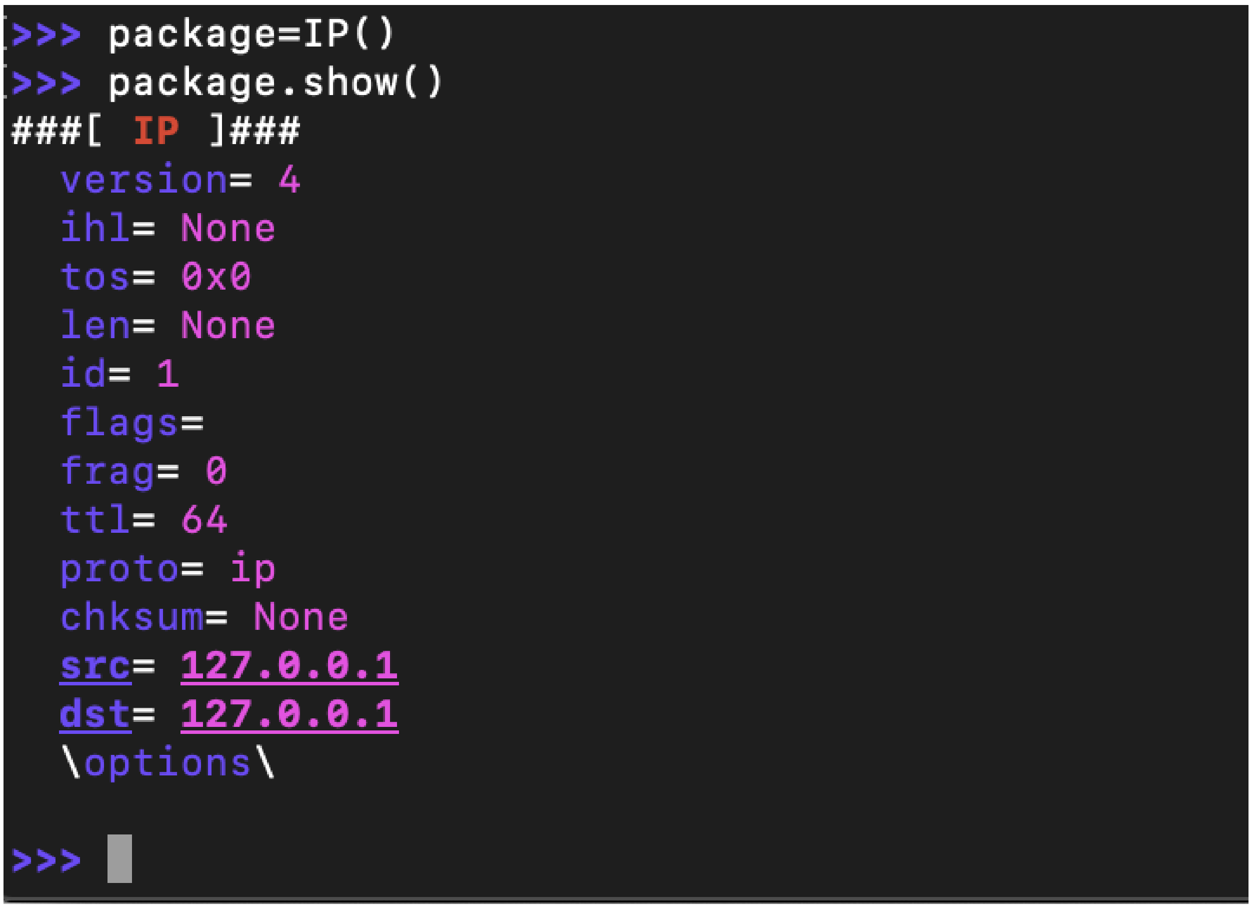
```
>>> lsc()
IPID_count          : Identify IP id values classes in a list of packets
arpcachepoison      : Poison target's cache with (your MAC,victim's IP) couple
arping              : Send ARP who-has requests to determine which hosts are up
arpleak             : Exploit ARP leak flaws, like NetBSD-SA2017-002.
bind_layers         : Bind 2 layers on some specific fields' values. It makes th
e packet being built  # noqa: E501
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 par
ameter
corrupt_bits        : Flip a given percentage or number of bits from a string
corrupt_bytes       : Corrupt a given percentage or number of bytes from a strin
g
defrag              : defrag(plist) -> ([not fragmented], [defragmented],
defragment          : defragment(plist) -> plist defragmented as much as possibl
e
dhcp_request        : Send a DHCP discover request and return the answer
dyndns_add          : Send a DNS add message to a nameserver for "name" to have
a new "rdata"
dyndns_del          : Send a DNS delete message to a nameserver for "name"
etherleak           : Exploit Etherleak flaw
explore             : Function used to discover the Scapy layers and protocols.
fletcher16_checkbytes: Calculates the Fletcher-16 checkbytes returned as 2 byte
binary-string.
fletcher16_checksum : Calculates Fletcher-16 checksum of the given buffer.
fragleak            : --
fragleak2           : --
fragment            : Fragment a big IP datagram
fuzz                : Transform a layer into a fuzzy layer by replacing some def
ault values by random objects
getmacbyip          : Return MAC address corresponding to a given IP address
getmacbyip6         : Returns the MAC address corresponding to an IPv6 address
hexdiff             : Show differences between 2 binary strings
hexdump             : Build a tcpdump like hexadecimal view
```

help()显示某一命令的使用帮助，如help(sniff)。



```
Help on function sniff in module scapy.sendrecv:

sniff(count=0, store=True, offline=None, prn=None, lfilter=None, L2socket=None,
timeout=None, opened_socket=None, stop_filter=None, iface=None, started_callback
=None, *arg, **karg)
    Sniff packets and return a list of packets.

    Args:
        count: number of packets to capture. 0 means infinity.
        store: whether to store sniffed packets or discard them
        prn: function to apply to each packet. If something is returned, it
             is displayed.
             --Ex: prn = lambda x: x.summary()
        filter: BPF filter to apply.
        lfilter: Python function applied to each packet to determine if
                 further action may be done.
                 --Ex: lfilter = lambda x: x.haslayer(Padding)
        offline: PCAP file (or list of PCAP files) to read packets from,
                 instead of sniffing them
        timeout: stop sniffing after a given time (default: None).
        L2socket: use the provided L2socket (default: use conf.L2listen).
        opened_socket: provide an object (or a list of objects) ready to use
                      .recv() on.
        stop_filter: Python function applied to each packet to determine if
                     we have to stop the capture after this packet.
                     --Ex: stop_filter = lambda x: x.haslayer(TCP)
        iface: interface or list of interfaces (default: None for sniffing
               on all interfaces).
        monitor: use monitor mode. May not be available on all OS
        started_callback: called as soon as the sniffer starts sniffing
                          (default: None).

    The iface, offline and opened_socket parameters can be either an
    element, a list of elements, or a dict object mapping an element to a
    label (see examples below).

    Examples:
      >>> sniff(filter="arp")
      >>> sniff(lfilter=lambda pkt: ARP in pkt)
:
```

show()显示指定数据包的详细信息。例如，这里我们先创建一个IP数据包，然后调用show方法。

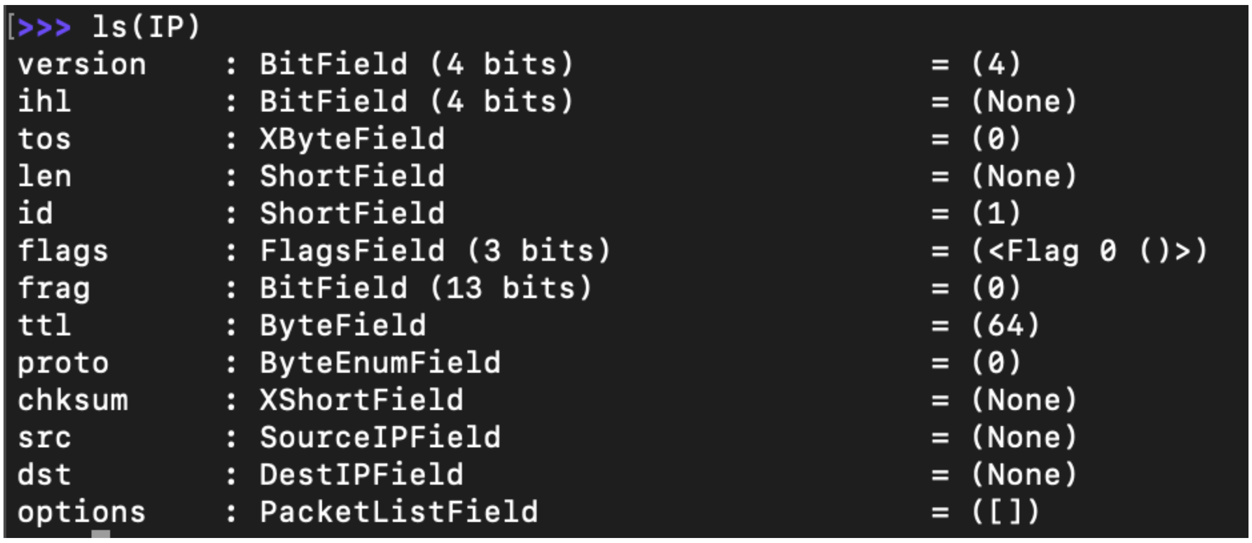


```
>>> package=IP()
>>> package.show()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= ip
  chksum= None
  src= 127.0.0.1
  dst= 127.0.0.1
  \options\

>>> 
```

3.4.3 综合练习

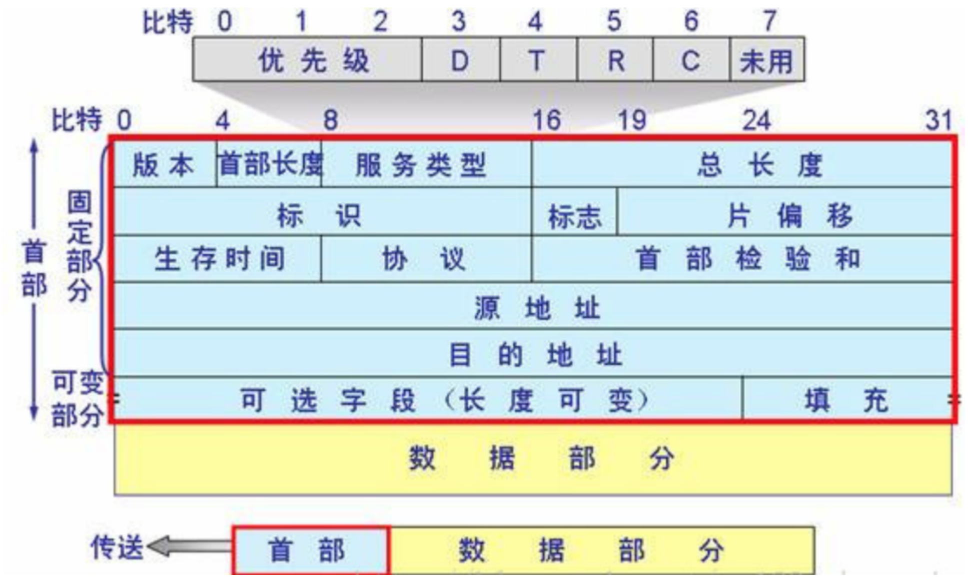
下面我们通过几个小例子，来加深对Scapy的理解。

我们可以使用Scapy来构造从数据链路层到应用层的任一层的数据包，需要各位同学参考不同协议的报文格式来练习。下面我构造一个IP数据包，先使用ls命令显示IP命令的参数。

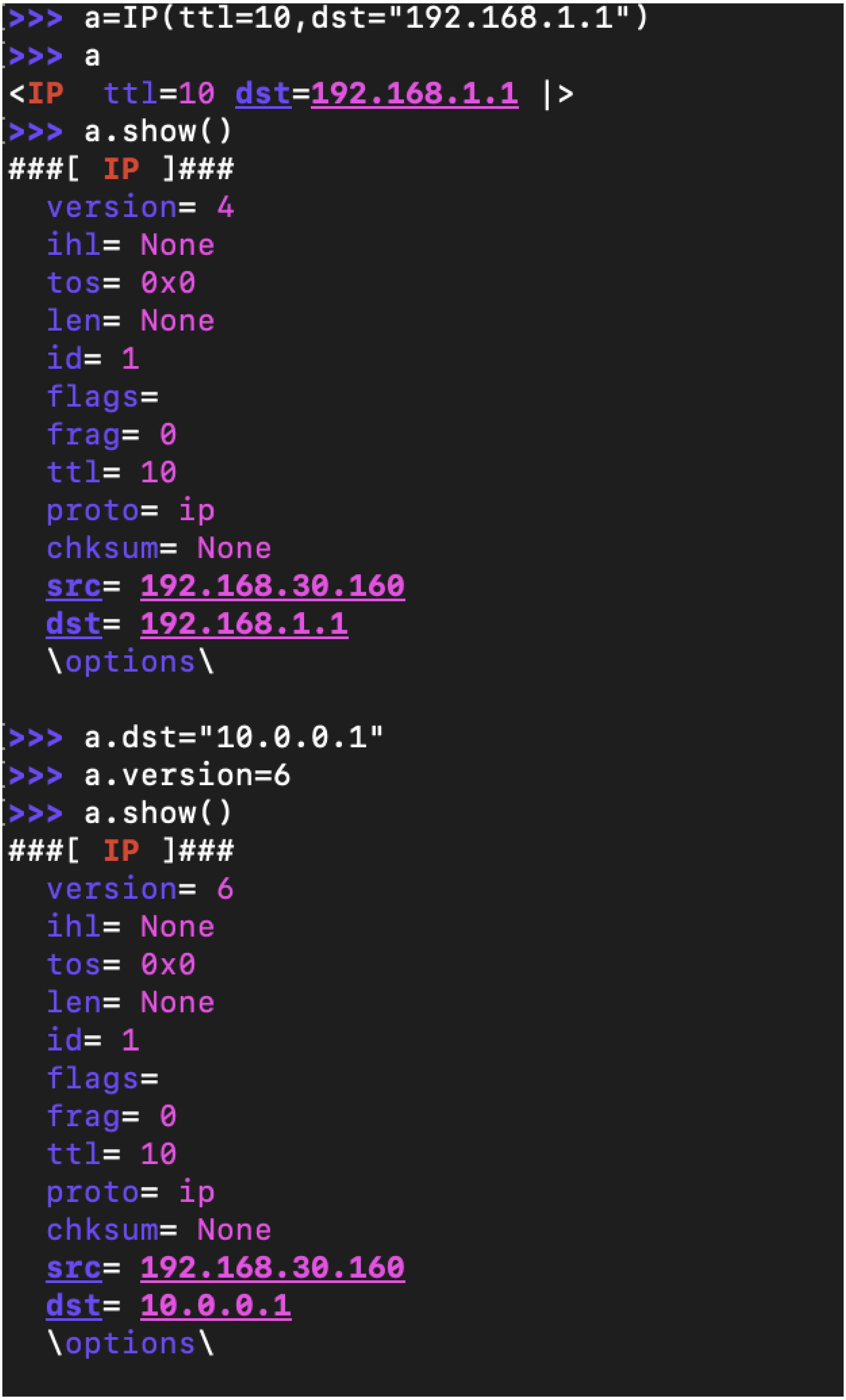


```
>>> ls(IP)
version    : BitField (4 bits)                    = (4)
ihl        : BitField (4 bits)                    = (None)
tos        : XByteField                           = (0)
len        : ShortField                           = (None)
id         : ShortField                           = (1)
flags      : FlagsField (3 bits)                  = (<Flag 0 ()>)
frag       : BitField (13 bits)                   = (0)
ttl        : ByteField                            = (64)
proto      : ByteEnumField                        = (0)
chksum     : XShortField                          = (None)
src        : SourceIPField                        = (None)
dst        : DestIPField                          = (None)
options    : PacketListField                      = ([])
```

每个字段是和IP协议一一对应的如下图：

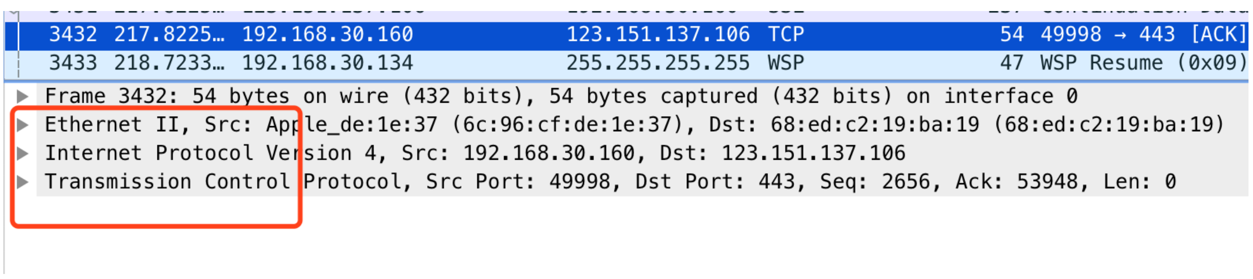


构造其他协议的数据包类似，只需要传入我们想要设置的值就可以了,返回的数据包对象可以再次修改。例如：



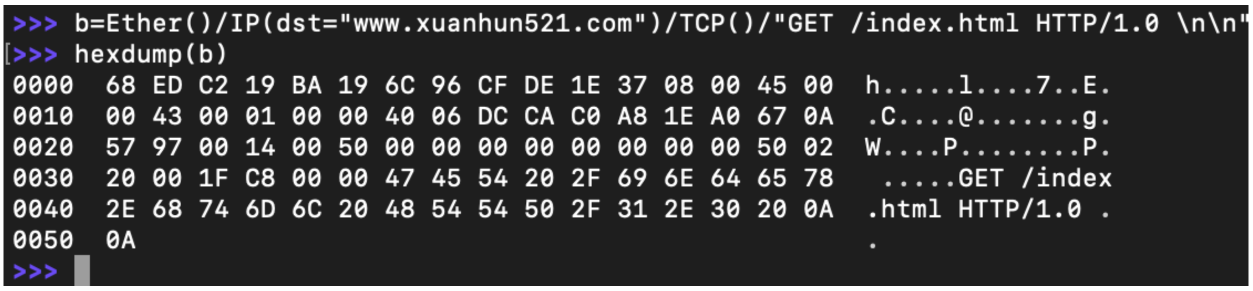
```
>>> a=IP(ttl=10,dst="192.168.1.1")
>>> a
<IP  ttl=10 dst=192.168.1.1 |>
>>> a.show()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 10
  proto= ip
  chksum= None
  src= 192.168.30.160
  dst= 192.168.1.1
  \options\

>>> a.dst="10.0.0.1"
>>> a.version=6
>>> a.show()
###[ IP ]###
  version= 6
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 10
  proto= ip
  chksum= None
  src= 192.168.30.160
  dst= 10.0.0.1
  \options\
```

因为网络数据包是层层包裹的，根据情况需要，也需要我们构建不同层的数据报文然后组合起来发送出去。使用"/"可以组合不同层的报文。 比如下面wireshark捕获的一个https报文：



```
3432  217.8225…  192.168.30.160      123.151.137.106  TCP       54  49998 → 443 [ACK]
3433  218.7233…  192.168.30.134      255.255.255.255  WSP       47  WSP Resume (0x09)
▶ Frame 3432: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
▶ Ethernet II, Src: Apple_de:1e:37 (6c:96:cf:de:1e:37), Dst: 68:ed:c2:19:ba:19 (68:ed:c2:19:ba:19)
▶ Internet Protocol Version 4, Src: 192.168.30.160, Dst: 123.151.137.106
▶ Transmission Control Protocol, Src Port: 49998, Dst Port: 443, Seq: 2656, Ack: 53948, Len: 0
```

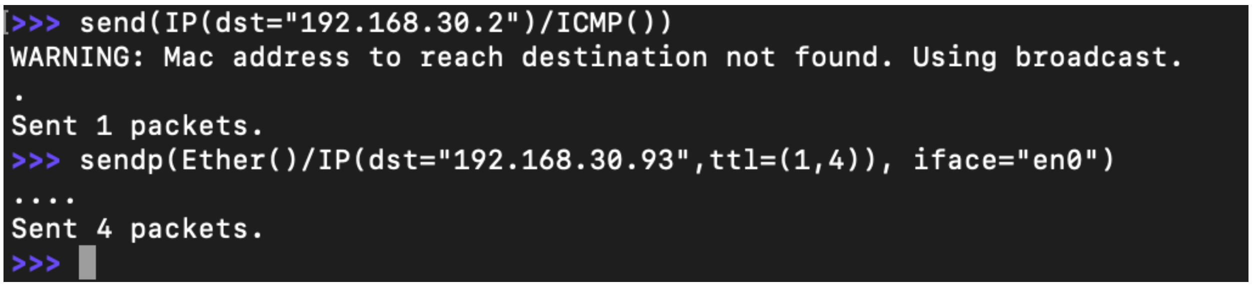
如果想从数据链路层将数据发送出去，就需要构造以太网帧数据，IP数据报文和TCP报文，并将三者组合起来发送出去。看下面的示例：



```
>>> b=Ether()/IP(dst="www.xuanhun521.com")/TCP()/"GET /index.html HTTP/1.0 \n\n"
>>> hexdump(b)
0000  68 ED C2 19 BA 19 6C 96 CF DE 1E 37 08 00 45 00  h.....l....7..E.
0010  00 43 00 01 00 00 40 06 DC CA C0 A8 1E A0 67 0A  .C....@.......g.
0020  57 97 00 14 00 50 00 00 00 00 00 00 00 00 50 02  W....P........P.
0030  20 00 1F C8 00 00 47 45 54 20 2F 69 6E 64 65 78   .....GET /index
0040  2E 68 74 6D 6C 20 48 54 54 50 2F 31 2E 30 20 0A  .html HTTP/1.0 .
0050  0A                                               .
>>>
```

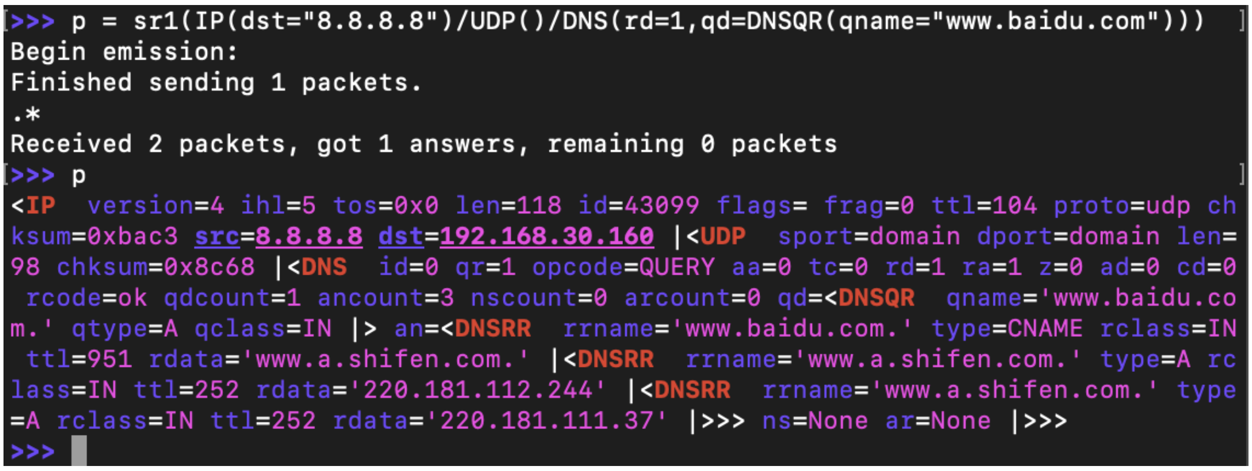
上图中我们使用了hexdump()函数， 使用hexdump()函数会以经典的hexdump格式输出数据包。

发送数据包可以使用的方法有两个send()和sendp()。send()函数将会在第3层发送数据包，也就是说它会为你处理路由和第2层的数据。sendp()函数将会工作在第2层。我们可以根据实际情况来决定使用哪个方法来发送数据。使用方法如下：



```
>>> send(IP(dst="192.168.30.2")/ICMP())
WARNING: Mac address to reach destination not found. Using broadcast.
.
Sent 1 packets.
>>> sendp(Ether()/IP(dst="192.168.30.93",ttl=(1,4)), iface="en0")
....
Sent 4 packets.
>>>
```

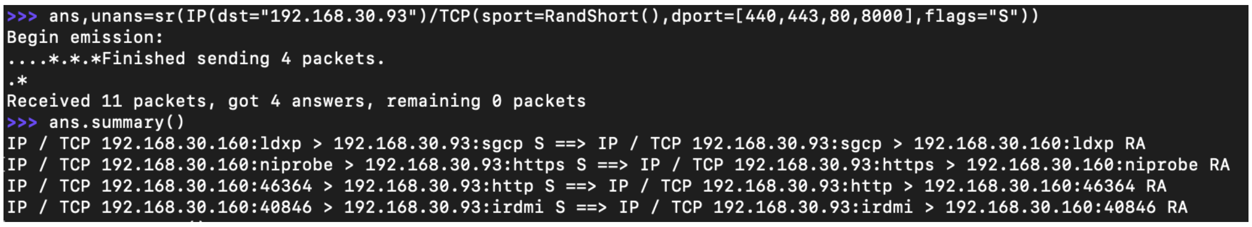
如果想要发送数据之后等待响应，可以使用sr()、sr1()或者srp()方法。sr()函数是用来发送数据包和接收应答。该函数返回一对数据包及其应答，还有无应答的数据包。sr1()函数是一种变体，用来返回一个应答数据包。发送的数据包必须是第3层报文（IP，ARP等）。srp()则是使用第2层报文（以太网，802.3等）。下面发送一个DNS查询的报文出去，接收查询结果。



```
>>> p = sr1(IP(dst="8.8.8.8")/UDP()/DNS(rd=1,qd=DNSQR(qname="www.baidu.com")))
Begin emission:
Finished sending 1 packets.
.*
Received 2 packets, got 1 answers, remaining 0 packets
>>> p
<IP  version=4 ihl=5 tos=0x0 len=118 id=43099 flags= frag=0 ttl=104 proto=udp ch
ksum=0xbac3 src=8.8.8.8 dst=192.168.30.160 |<UDP  sport=domain dport=domain len=
98 chksum=0x8c68 |<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 ancount=3 nscount=0 arcount=0 qd=<DNSQR  qname='www.baidu.co
m.' qtype=A qclass=IN |> an=<DNSRR  rrname='www.baidu.com.' type=CNAME rclass=IN
 ttl=951 rdata='www.a.shifen.com.' |<DNSRR  rrname='www.a.shifen.com.' type=A rc
lass=IN ttl=252 rdata='220.181.112.244' |<DNSRR  rrname='www.a.shifen.com.' type
=A rclass=IN ttl=252 rdata='220.181.111.37' |>>> ns=None ar=None |>>>
>>>
```

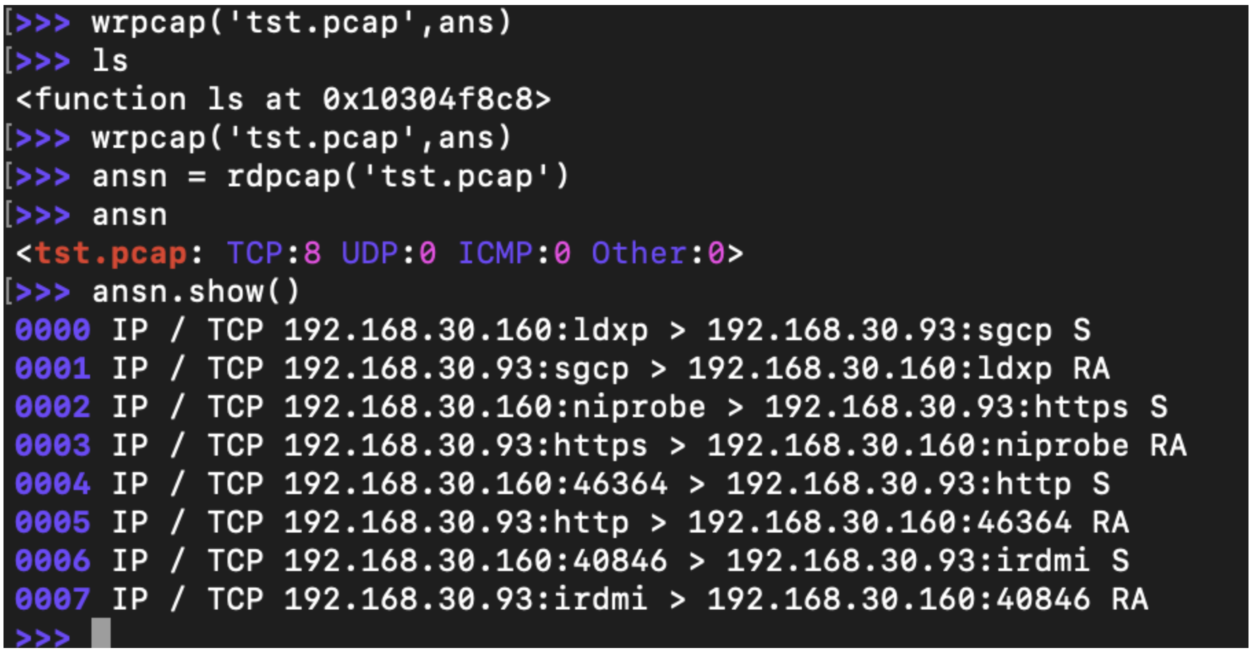
注意上图中我们使用了DNS()方法帮助构造应用层（DNS）的报文内容。

实际上接收的数据返回两个列表，第一个就是发送的数据包及其应答组成的列表，第二个是无应答数据包组成的列表。为了更好地呈现它们，它们被封装成一个对象，并且提供了一些便于操作的方法。 下面我们实现一个简单的SYN端口扫描：



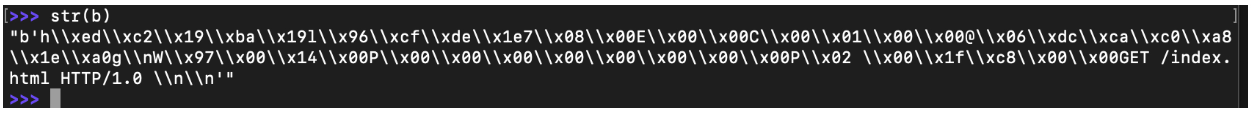
```
>>> ans,unans=sr(IP(dst="192.168.30.93")/TCP(sport=RandShort(),dport=[440,443,80,8000],flags="S"))
Begin emission:
....*.*.*Finished sending 4 packets.
.*
Received 11 packets, got 4 answers, remaining 0 packets
>>> ans.summary()
IP / TCP 192.168.30.160:ldxp > 192.168.30.93:sgcp S ==> IP / TCP 192.168.30.93:sgcp > 192.168.30.160:ldxp RA
IP / TCP 192.168.30.160:niprobe > 192.168.30.93:https S ==> IP / TCP 192.168.30.93:https > 192.168.30.160:niprobe RA
IP / TCP 192.168.30.160:46364 > 192.168.30.93:http S ==> IP / TCP 192.168.30.93:http > 192.168.30.160:46364 RA
IP / TCP 192.168.30.160:40846 > 192.168.30.93:irdmi S ==> IP / TCP 192.168.30.93:irdmi > 192.168.30.160:40846 RA
```

通常我们需要将数据包文件导出为pcap文件备用，需要的时候再导入，方法如下：



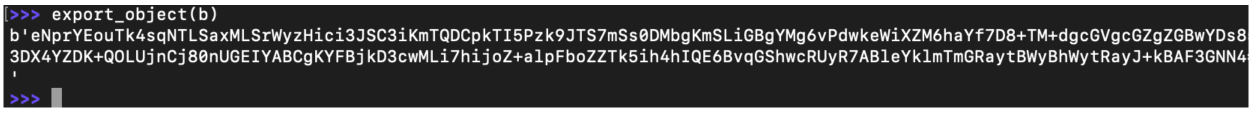
```
>>> wrpcap('tst.pcap',ans)
>>> ls
<function ls at 0x10304f8c8>
>>> wrpcap('tst.pcap',ans)
>>> ansn = rdpcap('tst.pcap')
>>> ansn
<tst.pcap: TCP:8 UDP:0 ICMP:0 Other:0>
>>> ansn.show()
0000 IP / TCP 192.168.30.160:ldxp > 192.168.30.93:sgcp S
0001 IP / TCP 192.168.30.93:sgcp > 192.168.30.160:ldxp RA
0002 IP / TCP 192.168.30.160:niprobe > 192.168.30.93:https S
0003 IP / TCP 192.168.30.93:https > 192.168.30.160:niprobe RA
0004 IP / TCP 192.168.30.160:46364 > 192.168.30.93:http S
0005 IP / TCP 192.168.30.93:http > 192.168.30.160:46364 RA
0006 IP / TCP 192.168.30.160:40846 > 192.168.30.93:irdmi S
0007 IP / TCP 192.168.30.93:irdmi > 192.168.30.160:40846 RA
>>>
```

使用str()函数可以将整个数据包转换成十六进制字符串：



```
>>> str(b)
"b'h\\xed\\xc2\\x19\\xba\\x19l\\x96\\xcf\\xde\\x1e7\\x08\\x00E\\x00\\x00C\\x00\\x01\\x00\\x00@\\x06\\xdc\\xca\\xc0\\xa8
\\x1e\\xa0g\\nW\\x97\\x00\\x14\\x00P\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00P\\x02 \\x00\\x1f\\xc8\\x00\\x00GET /index.
html HTTP/1.0 \\n\\n'"
>>>
```

使用export_object()函数，Scapy可以数据包转换成base64编码的Python数据结构：

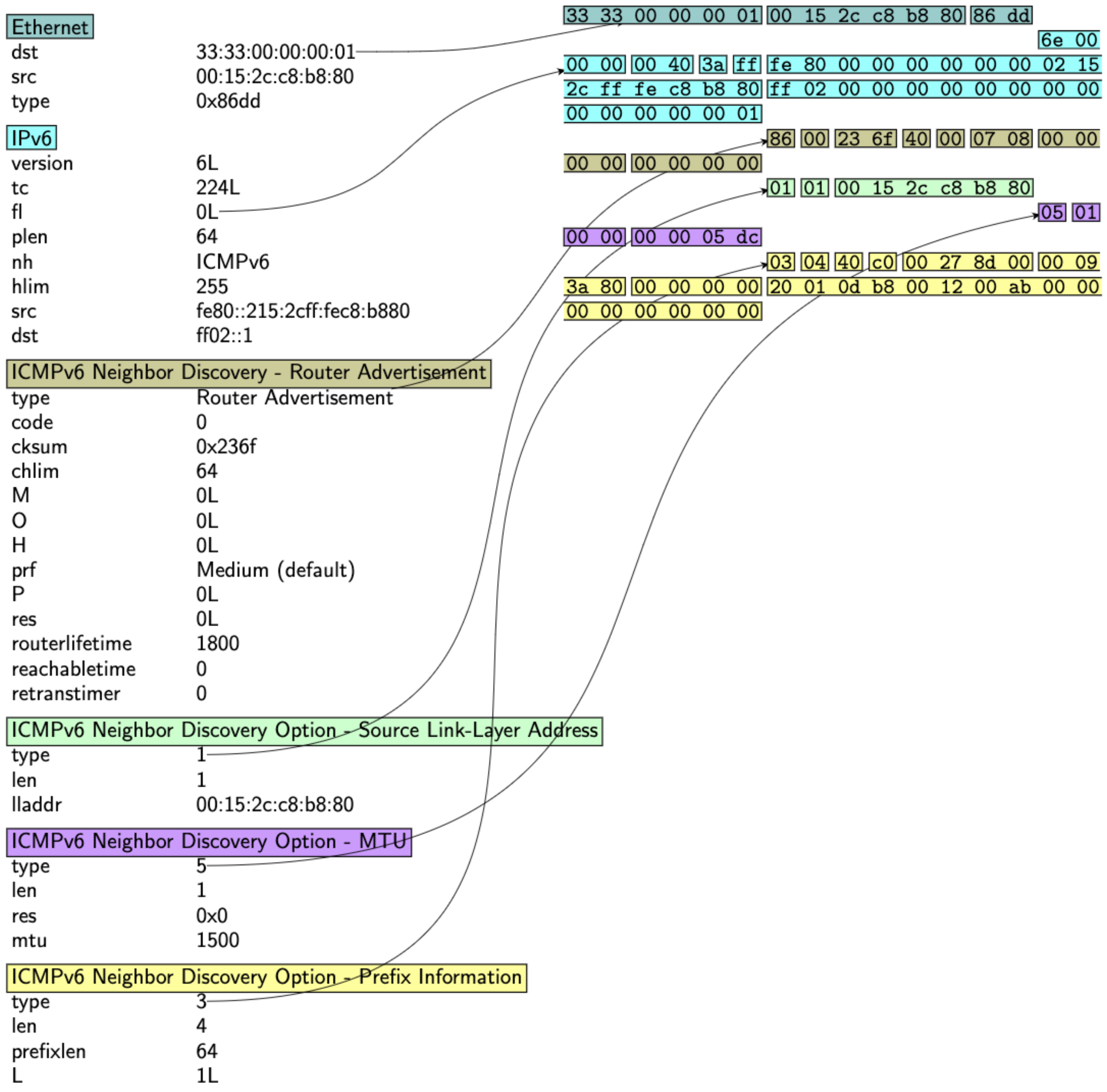


```
>>> export_object(b)
b'eNprYEouTk4sqNTLSaxMLSrWyzHici3JSC3iKmTQDCpkTI5Pzk9JTS7mSs0DMbgKmSLiGBgYMg6vPdwkeWiXZM6haYf7D8+TM+dgcGVgcGZgZGBwYDs8
3DX4YZDK+QOLUjnCj80nUGEIYABCgKYFBjkD3cwMLi7hijoZ+alpFboZZTk5ih4hIQE6BvqGShwcRUyR7ABleYklmTmGRaytBWyBhWytRayJ+kBAF3GNN4
'
>>> 
```

除此之外，如果您已经安装PyX，您可以做一个数据包的图形PostScript/ PDF转储,完整的输出命令列表如下：

命令	效果
str(pkt)	组装数据包

输出pdf示例如下：


Ethernet
dst 33:33:00:00:00:01
src 00:15:2c:c8:b8:80
type 0x86dd
IPv6
version 6L
tc 224L
fl 0L
plen 64
nh ICMPv6
hlim 255
src fe80::215:2cff:fec8:b880
dst ff02::1
ICMPv6 Neighbor Discovery - Router Advertisement
type Router Advertisement
code 0
cksum 0x236f
chlim 64
M 0L
O 0L
H 0L
prf Medium (default)
P 0L
res 0L
routerlifetime 1800
reachabletime 0
retranstimer 0
ICMPv6 Neighbor Discovery Option - Source Link-Layer Address
type 1
len 1
lladdr 00:15:2c:c8:b8:80
ICMPv6 Neighbor Discovery Option - MTU
type 5
len 1
res 0x0
mtu 1500
ICMPv6 Neighbor Discovery Option - Prefix Information
type 3
len 4
prefixlen 64
L 1L
33 33 00 00 00 01 00 15 2c c8 b8 80 86 dd
6e 00
00 00 00 40 3a ff fe 80 00 00 00 00 00 00 02 15
2c ff fe c8 b8 80 ff 02 00 00 00 00 00 00 00 00
00 00 00 00 00 01 86 00 23 6f 40 00 07 08 00 00
00 00 00 00 00 00 01 01 00 15 2c c8 b8 80 05 01
00 00 00 00 05 dc 03 04 40 c0 00 27 8d 00 00 09
3a 80 00 00 00 00 20 01 0d b8 00 12 00 ab 00 00
00 00 00 00 00 00

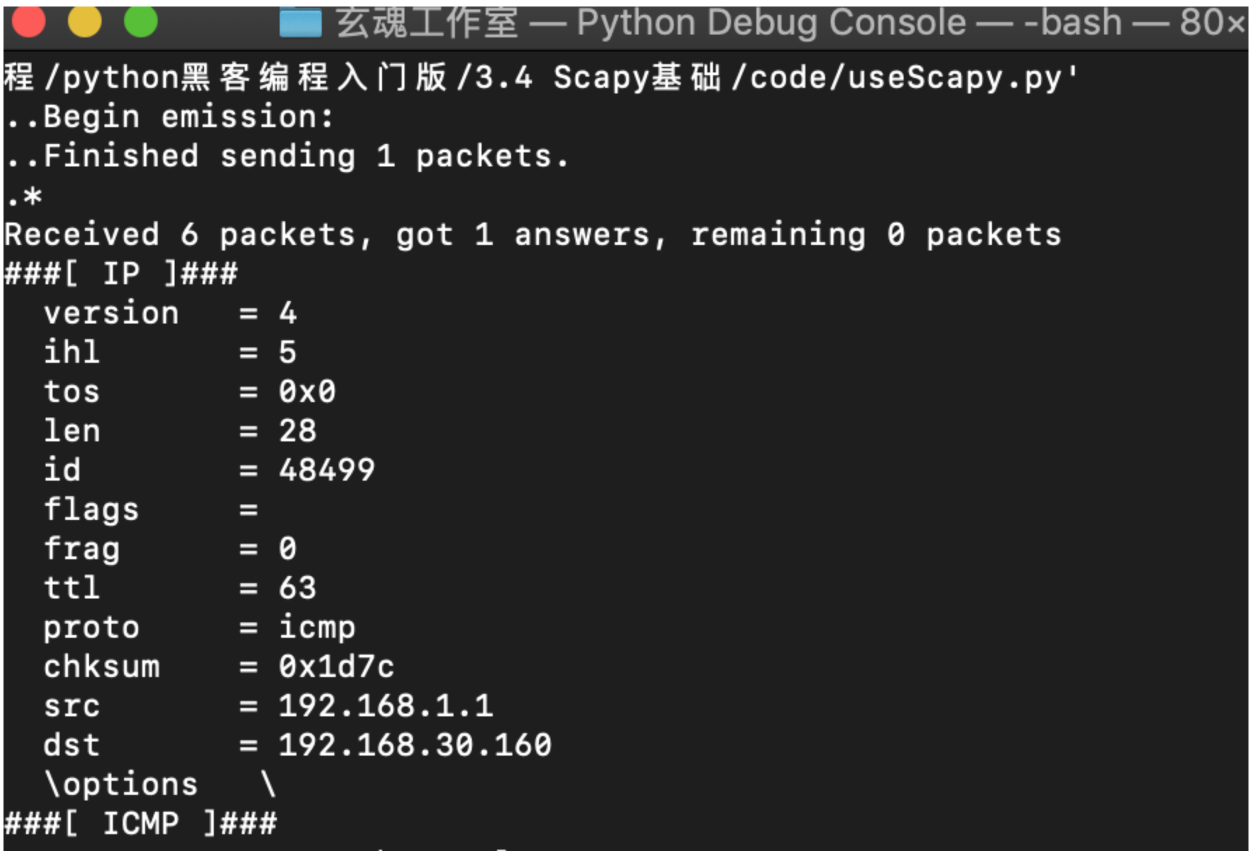
3.4.4 在Python中使用Scapy

在Python中调用Scapy很简单，只需要导入模块即可。

新建useScapy.py文件，添加如下代码：

```
# -*- coding: UTF-8 -*-

import sys
from scapy.all import *

p=sr1(IP(dst='192.168.1.1')/ICMP())
if p:
    p.show()
```

结果如下：



```
程/python黑客编程入门版/3.4 Scapy基础/code/useScapy.py'
..Begin emission:
..Finished sending 1 packets.
.*
Received 6 packets, got 1 answers, remaining 0 packets
###[ IP ]###
  version   = 4
  ihl       = 5
  tos       = 0x0
  len       = 28
  id        = 48499
  flags     =
  frag      = 0
  ttl       = 63
  proto     = icmp
  chksum    = 0x1d7c
  src       = 192.168.1.1
  dst       = 192.168.30.160
  \options   \
###[ ICMP ]###
```

3.4.5 小结

本节作为后面几个小节的前置知识，介绍了Scapy工具包的基本使用，更多的功能会在后面的章节继续介绍，同时建议各位同学阅读官方文档，全面了解。本节作业如下：

1. 安装Scapy
2. 属性基本的命令操作
3. 在Python中进行调用，实现ARP数据包的发送

下一节我们下沉到网络接口层，实现ARP欺骗工具。

> 欢迎到关注微信订阅号，交流学习中的问题和心得

玄魂工作室

本系列教程全部内容在玄说安全--入门圈发布，并提供答疑和辅导。