Raw socket programming in python (Linux)

Python By Silver Moon On Oct 9, 2012 22 Comments

Raw sockets allow a program or application to pr ovide custom headers for the specific pr otocol(tcp ip) which ar e otherwise pr ovided by the kernel/os network stack. In mor e simple terms its for adding custom headers instead of headers pr ovided by the underlying oper ating system.

Raw sock et support is available natively in the sock et api in linux. This is differ ent from windo ws where it is absent (it became available in windo ws 2000/xp/xp sp1 but was removed later). Although r aw sockets dont find much use in common networking applications, the y are used widely in applications r elated to network security .

In this article we ar e going to cr eate raw tcp/ip pack ets. For this we need to kno w how to make proper ip header and tcp headers. A pack et = Ip header + T cp header + data.

So lets have a look at the structur es.

Ip header

According to RFC 791

```
0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version|  IHL  |Type of Service|          Total Length         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Identification        |Flags|      Fragment Offset    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Time to Live |    Protocol   |         Header Checksum       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Source Address                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Destination Address                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Options                    |    Padding    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Every single number is 1 bit. So for e xample the V ersion field is 4 bit. The header must be constructed e xactly like shown.

TCP header

Next comes the T CP header. Accor ding to RFC 793

```
0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Source Port          |       Destination Port        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Sequence Number                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Acknowledgment Number                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Data |           |U|A|P|R|S|F|                               |
   | Offset| Reserved  |R|C|S|S|Y|I|            Window             |
   |       |           |G|K|H|T|N|N|                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           Checksum            |         Urgent Pointer        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Options                    |    Padding    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
    |                             data                              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```


Create a raw socket

Raw socket can be created in python like this

```
#create a raw socket
try:
        s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
except socket.error , msg:
        print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
        sys.exit()
```

To create raw socket, the program must have root privileges on the system. For example on ubuntu run the program with sudo. The above example creates a raw socket of type IPPROTO_RAW which is a raw IP packet. Means that we provide everything including the ip header.

Once the socket is created, next thing is to create and construct the packet that is to be send out. C like structures are not available in python, therefore the functions called pack and unpack have to be used to create the packet in the structure specified above.

So first, lets make the ip header

```
source_ip = '192.168.1.101'
dest_ip = '192.168.1.1' # or socket.gethostbyname('www.google.com')

# ip header fields
ip_ihl = 5
ip_ver = 4
ip_tos = 0
ip_tot_len = 0  # kernel will fill the correct total length
ip_id = 54321   #Id of this packet
ip_frag_off = 0
ip_ttl = 255
ip_proto = socket.IPPROTO_TCP
ip_check = 0    # kernel will fill the correct checksum
ip_saddr = socket.inet_aton ( source_ip )   #Spoof the source ip address if you want t
ip_daddr = socket.inet_aton ( dest_ip )

ip_ihl_ver = (version << 4) + ihl

# the ! in the pack format string means network order
ip_header = pack('!BBHHHBBH4s4s' , ip_ihl_ver, ip_tos, ip_tot_len, ip_id, ip_frag_off,
```

Now ip_header has the data for the ip header. Now the usage of pack function, it packs some values has bytes, some as 16bit fields and some as 32 bit fields.

Next comes the tcp header

```
# tcp header fields
tcp_source = 1234   # source port
tcp_dest = 80   # destination port
tcp_seq = 454
tcp_ack_seq = 0
tcp_doff = 5    #4 bit field, size of tcp header, 5 * 4 = 20 bytes
#tcp flags
tcp_fin = 0
tcp_syn = 1
tcp_rst = 0
tcp_psh = 0
tcp_ack = 0
tcp_urg = 0
tcp_window = socket.htons (5840)    #   maximum allowed window size
tcp_check = 0
tcp_urg_ptr = 0

tcp_offset_res = (tcp_doff << 4) + 0
tcp_flags = tcp_fin + (tcp_syn << 1) + (tcp_rst << 2) + (tcp_psh <<3) + (tcp_ack << 4)

# the ! in the pack format string means network order
tcp_header = pack('!HHLLBBHHH' , tcp_source, tcp_dest, tcp_seq, tcp_ack_seq, tcp_offse
```

The construction of the tcp header is similar to the ip header. The tcp header has a field called checksum which needs to be filled in corr ectly. A pseudo header is constructed to compute the checksum. The checksum is calculated over the tcp header along with the data. Checksum is necessar y to detect err ors in the tr ansmission on the r eceiver side.

Code

Here is the full code to send a r aw packet

```
'''
    Raw sockets on Linux

    Silver Moon (m00n.silv3r@gmail.com)
'''

# some imports
import socket, sys
from struct import *

# checksum functions needed for calculation checksum
def checksum(msg):
    s = 0

    # loop taking 2 characters at a time
    for i in range(0, len(msg), 2):
        w = ord(msg[i]) + (ord(msg[i+1]) << 8 )
        s = s + w

    s = (s>>16) + (s & 0xffff);
    s = s + (s >> 16);

    #complement and mask to 4 byte short
    s = ~s & 0xffff

    return s

#create a raw socket
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW)
except socket.error , msg:
    print 'Socket could not be created. Error Code : ' + str(msg[0]) + ' Message ' + 
    sys.exit()

# tell kernel not to put in headers, since we are providing it, when using IPPROTO_RA
# s.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)

# now start constructing the packet
packet = '';

source_ip = '192.168.1.101'
dest_ip = '192.168.1.1' # or socket.gethostbyname('www.google.com')

# ip header fields
ip_ihl = 5
ip_ver = 4
ip_tos = 0
ip_tot_len = 0  # kernel will fill the correct total length
ip_id = 54321   #Id of this packet
ip_frag_off = 0
ip_ttl = 255
ip_proto = socket.IPPROTO_TCP
ip_check = 0    # kernel will fill the correct checksum
ip_saddr = socket.inet_aton ( source_ip )   #Spoof the source ip address if you want 
ip_daddr = socket.inet_aton ( dest_ip )

ip_ihl_ver = (ip_ver << 4) + ip_ihl

# the ! in the pack format string means network order
ip_header = pack('!BBHHHBBH4s4s' , ip_ihl_ver, ip_tos, ip_tot_len, ip_id, ip_frag_off

# tcp header fields
tcp_source = 1234   # source port
tcp_dest = 80   # destination port
tcp_seq = 454
tcp_ack_seq = 0
tcp_doff = 5    #4 bit field, size of tcp header, 5 * 4 = 20 bytes
#tcp flags
tcp_fin = 0
tcp_syn = 1
tcp_rst = 0
tcp_psh = 0
tcp_ack = 0
tcp_urg = 0
tcp_window = socket.htons (5840)    #   maximum allowed window size
tcp_check = 0
tcp_urg_ptr = 0

tcp_offset_res = (tcp_doff << 4) + 0
tcp_flags = tcp_fin + (tcp_syn << 1) + (tcp_rst << 2) + (tcp_psh <<3) + (tcp_ack << 4

```

```
# the ! in the pack format string means network order
tcp_header = pack('!HHLLBBHHH' , tcp_source, tcp_dest, tcp_seq, tcp_ack_seq, tcp_offs

user_data = 'Hello, how are you'

# pseudo header fields
source_address = socket.inet_aton( source_ip )
dest_address = socket.inet_aton(dest_ip)
placeholder = 0
protocol = socket.IPPROTO_TCP
tcp_length = len(tcp_header) + len(user_data)

psh = pack('!4s4sBBH' , source_address , dest_address , placeholder , protocol , tcp_
psh = psh + tcp_header + user_data;

tcp_check = checksum(psh)
#print tcp_checksum

# make the tcp header again and fill the correct checksum - remember checksum is NOT
tcp_header = pack('!HHLLBBH' , tcp_source, tcp_dest, tcp_seq, tcp_ack_seq, tcp_offset_

# final full packet - syn packets dont have any data
packet = ip_header + tcp_header + user_data

#Send the packet finally - the port specified has no effect
s.sendto(packet, (dest_ip , 0 ))    # put this in a loop if you want to flood the tar
```

Run the above program from the terminal and check the network tr affic using a packet sniffer like wireshark. It should show the packet.

Raw sockets find application in the field of network security . The above example can be used to code a tcp syn flood program. Syn flood programs are used in Dos attacks. Raw sock ets are also used to code pack et sniffers, port scanners etc.

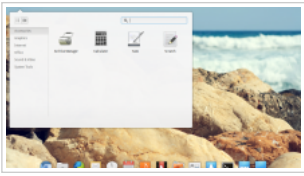
Last Updated On : 11th October 2012

python python sockets raw sockets

Related Posts

Syn flood program in python using raw sockets (Linux)

Python socket – network programming tutorial



Programming udp sockets in python

About **Silver Moon**

Php developer, blogger and Linux enthusiast. He can be reached at binarytides@gmail.com. Or find him on Google+

22 Comments

+ ADD COMMENT

Jhon June 8, 2016 at 11:13 pm

The spoofed ip dosnt show up on wireshark.

visitor April 7, 2016 at 2:26 am

You really need a better comment posting mechanism
that allows your visitors to easily post python code
without destroying the indentation.

I'm really really disgusted, and i would not trust your
code with a monkey I did not like.

visitor April 7, 2016 at 2:24 am

Last try: code tag and a line start marker to try to preserve spaces:

```
! def checksum(msg):
! vals=map(ord,msg)
! if ((len(vals)>>1)<<1) != len(vals): vals.append(0) # odd length
! s=sum([vals[i]+(vals[i+1]< 0xffff: s=(s>>16)+(s & 0xffff)
! return s ^ 0xffff # ones complement
```

visitor April 7, 2016 at 2:21 am

OK, let's try the code tag.

```
def checksum(msg):
vals=map(ord,msg)
if ((len(vals)>>1)<<1) != len(vals): vals.append(0) # odd length
s=sum([vals[i]+(vals[i+1]< 0xffff: s=(s>>16)+(s & 0xffff)
return s ^ 0xffff # ones complement
```

visitor April 7, 2016 at 2:18 am

Fix indentation and code mangling by your crappy posting system:

> *> def checksum(msg):*
> *> vals=map(ord,msg)*
> *> if ((len(vals)>>1)< s=sum([vals[i]+(vals[i+1]< while s > oxffff: s=(s>>16)+(s & oxffff)*
> *> return s ^ oxffff # ones complement*

visitor April 7, 2016 at 2:08 am

Simplified, clarified, hardened, and corrected(?) checksum function:

def checksum(msg):
vals=map(ord,msg)
if ((len(vals)>>1)<<1) != len(vals): vals.append(0) # odd length
s=sum([vals[i]+(vals[i+1]< 0xffff: s=(s>>16)+(s & 0xffff)
return s ^ 0xffff # unsigned ones complement

This assumes the proper checksum is defined as a 16 bit sum reduction
followed by a bitwise complement. The original code is buggy.

1) It fails to prevent an indexing error for odd length messages.
2) It fails to consistently compute the checksum for very long messages.
3) Its 'ones complement' is an integer signed ones complement, not an unsigned binary complement.

If the rest of the code is as bad, user beware. One useful thing I did learn
was the necessity of administrator privilege for creating this socket.

jeff March 1, 2016 at 2:25 pm

I don't understand your TCP checksum function, at the end, you can simplify it with only ;

s = s + (s >> 16) # add the carry to the result
s = ~s & 0xffff # one complement and mask to 4 byte short

This line isn't useful : s = (s>>16) + (s & 0xffff)

mat January 9, 2015 at 5:41 pm

why do you compute the checksum yourself when in your comments you state that the kernel does this for us?

aaron November 9, 2014 at 4:15 am

I am trying to captures these packets in wireshark after running the above code by putting the filet ip.src == 192.168.1.101. But I don't see any packets. I changed the destination address in the above code to my PC IP address.

Aaron November 9, 2014 at 3:28 am

I executed this code. I changed the destination IP to my system IP. I'm trying to capture packets in wireshark by filtering ip.src ==192.168.1.101 but I'm not receiving any packets.

aaron October 21, 2014 at 1:47 am

Is it possible to send ICMP using the same socket? OR do we need to replace IPPROTO_RAW with IPPROTO_ICMP_TCP? Also is it the same way we receive the raw socket?

Patrick Leedom June 4, 2013 at 9:38 pm

Why does the Ethernet frame not have any MAC addresses? I thought the kernel handled all of the ethernet headers.

Silver Moon June 5, 2013 at 7:32 am

where is the mac address missing ? In the packets send out by the python program ?

Patrick Leedom June 5, 2013 at 9:43 am

I was sending it to loopback and the kernel doesnt add a mac since it doesnt go through a nic. My mistake.

holia April 26, 2013 at 8:39 pm

What is the rule of ! in pact like BBH4S ? How could we define one ? thanks .

holia April 26, 2013 at 8:35 pm

How to define flag for ip header ? for example flag for "more fragment" . thanks .

Jacobson April 25, 2013 at 3:17 pm

This is great man.thanks. It could be use in making firewalls crazy :) .

Silver Moon April 25, 2013 at 3:25 pm

modern firewalls are very well configured.
they would even block a host that sends too many invalid packets.

demplers December 19, 2012 at 4:52 pm

How to use UDP Raw Socket?

Silver Moon December 19, 2012 at 4:59 pm

follow this article
http://www.binarytides.com/raw-udp-sockets-c-linux/

demplers December 19, 2012 at 5:11 pm

this is a C!!!!
How to use UDP Raw Socket in Python?

gforcelx January 22, 2013 at 6:25 pm

I also want to know!! UDP Raw Socket... can you help? thanks

Leave a comment

Name (required)

Mail (will not be published) (r equired)

Website

Comment

POST COMMENT