

Forum:[Hack Forums](#)

Thread: [Raw socket not reading syn-ack response](#)

moproblems99 - April 13, 2018, 3:33 pm

Greetings everyone,

I decided to start learning some Python and its related OOP implementation so I started to write this port scanner. Right now it is really just a PoC so all addresses and what not are hard coded. The main problem that I am having is that the program is not reading in the syn-ack packet returned by the target. I can see the initial syn packet being created by my program and the resulting syn-ack from the target in Wireshark which is leading me to believe that my syn packet is correctly formatted.

My program sits at line waiting to receive a packet and never receives the packet although I know it is there. I had to cut out some of the debugging code I had in there to fit in the post so hopefully the code works as I was expecting.

Program Architecture

Project Directory

networking

 __init__.py

 ipheader.py

 packet.py

 tcpheader.py

PythonScanner.py

Main Test Program

Code import socket, sys

from networking.packet import IPPacket

transmitter = None;

receiver = None;

source_ip = '10.0.2.15';

destination_ip = '192.168.0.11';

source_port = 1337;

destination_port = 80

packet = IPPacket();

packet.set_tcpheader(source_port, destination_port, [0,1,0,0,0,0]);

print('Packed TCP Header: ');

print(packet.get_packed_tcpheader());

packet.set_ipheader(socket.IPPROTO_TCP, source_ip, destination_ip);

print('Packed IP Header: ');

print(packet.get_packed_ipheader());

```

print('Packet ready to send: ');
print(packet.get_network_output());

#create a raw socket
print('Creating transmitter...');

try:
    transmitter = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_RAW);
    print('Transmitter created...');
    # transmitter.bind(('localhost', source_port));
except socket.error as error:
    print('Transmitter could not be created. Error Code: ');
    print(error);
    sys.exit();

#Send the packet
result = transmitter.sendto(packet.get_network_output(), (destination_ip , source_port ))
print('Packet sent!');
result, address = transmitter.recvfrom(65535);

```

```

print('Received from ' + str(destination_ip) + ': ');
print(result);
Networking Package __init__.py

```

Packet Class

Code # CClass Definition of Packet

```

import socket,sys
from struct import pack
from networking.tcpheader import TCPHeader
from networking.ipheader import IPHeader

```

```

class IPPacket:
    """Class for IP Packet"""

    tcp_header = None;
    ip_header = None;
    data = "";
    tcp_length = 0;

    def __init__(self):
        pass

    #Packet methods
    def calc_checksum(self, packet):
        output = 0

```

```

for i in range(0, len(packet), 2):
    word = packet[i] + (packet[i+1] << 8 ) #word = ord(packet[i]) + (ord(packet[i+1]) << 8 )
    output = output + word

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

output = ~output & 0xffff

#print('checksum debug: '+str(output));

return output

def get_network_output(self):
    self.tcp_length = len(self.data) + len(self.get_packed_tcpheader());

    psh = pack('!4s4sBBH', self.ip_header.source_ip , self.ip_header.dest_ip , 0 ,
self.ip_header.protocol , self.tcp_length);
    psh = psh + self.get_packed_tcpheader() + self.data.encode('ascii');

    tcp_checksum = self.calc_checksum(psh);

    self.set_tcpchecksum(tcp_checksum);

    return self.get_packed_ipheader() + self.get_packed_tcpheader() + self.data.encode('ascii');

def build_checksum_string(self):
    self.tcp_length = len(self.data) + len(self.get_packed_tcpheader());

    psh = pack('!4s4sBBH', self.ip_header.source_ip , self.ip_header.dest_ip , 0 ,
self.ip_header.protocol , self.tcp_length);
    psh = psh + self.get_packed_tcpheader() + self.data.encode('ascii');

    return psh;

#TCP methods
def set_tcpheader(self, source, dest, flags):
    self.tcp_header = TCPHeader(source, dest, flags);

def get_packed_tcpheader(self):
    return(self.tcp_header.pack_header());

def set_tcpchecksum(self, val):
    self.tcp_header.set_checksum(val);

#IP methods

def set_ipheader(self, protocol, source, dest):
    self.ip_header = IPHeader(protocol, source, dest);

```

```

def get_packed_ipheader(self):
    return(self.ip_header.pack_header());

# data methods
TCP Header
Code # Class file for TCP header
import socket,sys
from struct import pack
from collections import OrderedDict, namedtuple

class TCPHeader():

    #need to do flags properly

    FLAGS = namedtuple('TCPFlags', 'urg ack psh rst syn fin')(fin = 1, syn = 2, rst = 4, psh = 8, ack =
16, urg = 32);

    source_port = None;
    dest_port = None;
    seq = None;
    ack = None;
    data_offset = None;
    # flags = {'fin'=0, 'syn'=0, 'rst'=0, 'psh'=0, 'ack'=0, 'urg'=0};
    flags = 0;
    window = None;
    check = None;
    urg_ptr = None;
    offset_res = None;

def __init__(self, source, dest, flags):
    self.source_port = source;
    self.dest_port = dest;
    self.seq = 1337;
    self.ack = 1337;
    self.data_offset = 5;
    self.flags = self.build_flags(flags);
    self.window = socket.htons(5840);
    self.check = 0;
    self.urg_ptr = 0;
    self.offset_res = (self.data_offset << 4) + 0

def build_flags(self, flags):

    # needs:
    # validation
    # cleansing

    # return var
    f = 0

    if isinstance(flags, list):
        for i, val in enumerate(flags):

```

```

        f += val << i;
    else:
        f = flags;

    return f;

def set_checksum(self, val):
    self.check = val

def pack_header(self):

    return pack('!HHLLBBHHH' ,
        self.source_port,
        self.dest_port,
        self.seq,
        self.ack,
        self.offset_res,
        self.flags,
        self.window,
        socket.htons(self.check),
        self.urg_ptr);

```

IP Header

Code # Class file for IP header

```

import socket,sys
from struct import pack

```

```

class IPHeader():
    ihl = None;
    ver = None;
    tos = None;
    tot_len = None;
    id = None;
    frag_off = None;
    ttl = None;
    protocol = None;
    check = None;
    source_ip = None;
    dest_ip = None;
    ihl_ver = None;

    def __init__(self, proto, source, dest):
        self.ihl = 5;
        self.ver = 4;
        self.tos = 0;
        self.tot_len = 0;
        self.id = 1337;
        self.frag_off = 0;
        self.ttl = 255;
        self.protocol = proto;
        self.check = 0;

```

```
self.ihl_ver = (self.ver << 4) + self.ihl;  
self.source_ip = socket.inet_aton(source);  
self.dest_ip = socket.inet_aton(dest);
```

```
def pack_header(self):  
    return pack(  
        '!BBHHHBBH4s4s' ,  
        self.ihl_ver,  
        self.tos,  
        self.tot_len,  
        self.id,  
        self.frag_off,  
        self.ttl,  
        self.protocol,  
        self.check,  
        self.source_ip,  
        self.dest_ip);
```