############# Application #3 - Part #1 #############

#DHCP client simulator

#In scapy interactive mode - DHCP packets:

'''

'Ether / IP / UDP 0.0.0.0:bootpc > 255.255.255.255:bootps / BOOTP / DHCP'

DHCP DISCOVER:

'Ether(src=\'08:00:27:f9:51:87\', dst=\'ff:ff:ff:ff:ff:ff\', type=2048)/IP(frag=0L, src=\'0.0.0.0\', proto=17, tos=16, dst=\'255.255.255.255\', chksum=14742, len=328, options=[], version=4L, flags=0L, ihl=5L, ttl=128, id=0)/UDP(dport=67, sport=68, len=308, chksum=47898)/BOOTP(hlen=6, sname=\'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\', xid=398202904, ciaddr=\'0.0.0.0\', hops=0, giaddr=\'0.0.0.0\', chaddr="\\x08\\x00\'\\xf9Q\\x87\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00", yiaddr=\'0.0.0.0\', secs=0, flags=0L, htype=1, file=\'\\x00\', siaddr=\'0.0.0.0\', options=\'c\\x82Sc\', op=1)/DHCP(options=[(\'message-type\', 1), (\'hostname\', \'kali-teo\'), (\'param_req_list\', \'\\x01\\x1c\\x02\\x03\\x0f\\x06w\\x0c,/\\x1ay*\'), \'end\', \'pad\'])'

DHCP OFFER:

'Ether(src=\'c0:04:1a:5c:00:01\', dst=\'08:00:27:f9:51:87\', type=2048)/IP(frag=0L, src=\'192.168.2.111\', proto=17, tos=0, dst=\'192.168.2.1\', chksum=13540, len=328, options=[], version=4L, flags=0L, ihl=5L, ttl=255, id=0)/UDP(dport=68, sport=67, len=308, chksum=19350)/BOOTP(hlen=6, sname=\'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\', xid=398202904, ciaddr=\'0.0.0.0\', hops=0, giaddr=\'0.0.0.0\', chaddr="\\x08\\x00\'\\xf9Q\\x87\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00", yiaddr=\'192.168.2.1\', secs=0, flags=0L, htype=1, file=\'\\x00\', siaddr=\'0.0.0.0\', options=\'c\\x82Sc\', op=2)/DHCP(options=[(\'message-type\', 2), (\'server_id\', \'192.168.2.111\'), (\'lease_time\', 86400), (\'renewal_time\', 43200), (\'rebinding_time\', 75600), (\'subnet_mask\', \'255.255.255.0\'), \'end\', \'pad\'])'

DHCP OFFER (more options):

'Ether(src='ca:04:15:ec:00:08', dst='00:00:5e:4a:a3:fe', type=2048)/IP(frag=0L, src='192.168.2.111', proto=17, tos=0, dst='192.168.2.236', chksum=9573, len=328, options=[], version=4L, flags=0L, ihl=5L, ttl=255, id=3732)/UDP(dport=68, sport=67, len=308, chksum=3558)/BOOTP(hlen=6, sname='\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00', xid=868370, ciaddr='0.0.0.0', hops=0, giaddr='0.0.0.0', chaddr='\\x00\\x00^J\\xa3\\xfe\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00', yiaddr='192.168.2.236', secs=0, flags=0L, htype=1, file='\\x00', siaddr='0.0.0.0',

options='c\\x82Sc', op=2)/DHCP(options=[('message-type', 2), ('server_id', '192.168.2.111'), ('lease_time', 86400), ('renewal_time', 43200), ('rebinding_time', 75600), ('subnet_mask', '255.255.255.0'), ('router', '192.168.2.254'), 'end', 'pad'])'

DHCP REQUEST:

'Ether(src=\'08:00:27:f9:51:87\', dst=\'ff:ff:ff:ff:ff:ff\', type=2048)/IP(frag=0L, src=\'0.0.0.0\', proto=17, tos=16, dst=\'255.255.255.255\', chksum=14742, len=328, options=[], version=4L, flags=0L, ihl=5L, ttl=128, id=0)/UDP(dport=67, sport=68, len=308, chksum=61228)/BOOTP(hlen=6, sname=\'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\', xid=398202904, ciaddr=\'0.0.0.0\', hops=0, giaddr=\'0.0.0.0\', chaddr="\\x08\\x00\'\\xf9Q\\x87\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00", yiaddr=\'0.0.0.0\', secs=0, flags=0L, htype=1, file=\'\\x00\', siaddr=\'0.0.0.0\', options=\'c\\x82Sc\', op=1)/DHCP(options=[(\'message-type\', 3), (\'server_id\', \'192.168.2.111\'), (\'requested_addr\', \'192.168.2.1\'), (\'hostname\', \'kali-teo\'), (\'param_req_list\', \'\\x01\\x1c\\x02\\x03\\x0f\\x06w\\x0c,/\\x1ay*\'), \'end\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\'])'

DHCP ACK:

'Ether(src=\'c0:04:1a:5c:00:01\', dst=\'08:00:27:f9:51:87\', type=2048)/IP(frag=0L, src=\'192.168.2.111\', proto=17, tos=0, dst=\'192.168.2.1\', chksum=13539, len=328, options=[], version=4L, flags=0L, ihl=5L, ttl=255, id=1)/UDP(dport=68, sport=67, len=308, chksum=18582)/BOOTP(hlen=6, sname=\'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\', xid=398202904, ciaddr=\'0.0.0.0\', hops=0, giaddr=\'0.0.0.0\',

```
chaddr="\\x08\\x00\'\\xf9Q\\x87\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00",
yiaddr=\'192.168.2.1\', secs=0, flags=0L, htype=1,
file=\'\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00
\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\\x00\', siaddr=\'0.0.0.0\',
options=\'c\\x82Sc\', op=2)/DHCP(options=[(\'message-type\', 5), (\'server_id\', \'192.168.2.111\'),
(\'lease_time\', 86400), (\'renewal_time\', 43200), (\'rebinding_time\', 75600), (\'subnet_mask\',
\'255.255.255.0\'), \'end\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\',
\'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\', \'pad\',
\'pad\', \'pad\', \'pad\', \'pad\'])'

'''

import subprocess

import logging

import random

import sys

#This will suppress all messages that have a lower level of seriousness than error messages, while running or loading Scapy

logging.getLogger("scapy.runtime").setLevel(logging.ERROR)

logging.getLogger("scapy.interactive").setLevel(logging.ERROR)

logging.getLogger("scapy.loading").setLevel(logging.ERROR)

try:
```

```
    from scapy.all import *


except ImportError:
    print "Scapy package for Python is not installed on your system."
    print "Get it from https://pypi.python.org/pypi/scapy and try again."
    sys.exit()




#To see a list of what commands Scapy has available, run the lsc() function.
#Run the ls() command to see ALL the supported protocols.
#Run the ls(protocol) command to see the fields and default values for any protocol.
#See packet layers with the .summary() function.
#See packet contents with the .show() function.
#Dig into a specific packet layer using a list index: pkts[3][2].summary()...
#...the first index chooses the packet out of the pkts list, the second index chooses the layer for that specific packet.
#Using the .command() packet method will return a string of the command necessary to recreate that sniffed packet.




print "\n! Make sure to run this program as ROOT !\n"


#Setting network interface in promiscuous mode
net_iface = raw_input("Enter the interface to the target network: ")
```

```
subprocess.call(["ifconfig", net_iface, "promisc"], stdout=None, stderr=None, shell=False)


print "\nInterface %s was set to PROMISC mode." % net_iface



#Scapy normally makes sure that replies come from the same IP address the stimulus was sent to.

#But our DHCP packet is sent to the IP broadcast address (255.255.255.255) and any answer packet will have the IP address of the replying DHCP server as its source IP address (e.g. 192.168.2.101).

#Because these IP addresses don't match, we have to disable Scapy's check with conf.checkIPaddr = False before sending the stimulus.

#Source: https://bitbucket.org/pbi/test/wiki/doc/IdentifyingRogueDHCPServers

conf.checkIPaddr = False



   ############## Application #3 - Part #2 ##############



################### DHCP SEQUENCE ##################

all_given_leases = []

server_id = []

client_mac = []


#Generate entire DHCP sequence

def generate_dhcp_seq():
```

```
global all_given_leases


#Defining some DHCP parameters

x_id = random.randrange(1, 1000000)

hw = "00:00:5e" + str(RandMAC())[8:]

hw_str = mac2str(hw)

#print hw


#Assigning the .command() output of a captured DHCP DISCOVER packet to a variable

dhcp_dis_pkt = Ether(dst="ff:ff:ff:ff:ff:ff", src=hw)/IP(src="0.0.0.0",dst="255.255.255.255") /
UDP(sport=68,dport=67)/BOOTP(op=1, xid=x_id, chaddr=hw_str)/DHCP(options=[("message-
type","discover"),("end")])


#Sending the DISCOVER packet and catching the OFFER reply

#Generates two lists (answ and unansw). answd is a list containg a tuple: the first element is the
DISCOVER packet, the second is the OFFER packet

answd, unanswd = srp(dhcp_dis_pkt, iface=pkt_inf, timeout = 2.5, verbose=0)


#print answd

#print unanswd

#print answd.summary()

#print unanswd.summary()

#print answd[0][1][BOOTP].yiaddr


#The IP offered by the DHCP server to the client is extracted from the received answer

offered_ip = answd[0][1][BOOTP].yiaddr

#print offered_ip
```

```
    #Assigning the .command() output of a captured DHCP REQUEST packet to a variable

    dhcp_req_pkt = Ether(dst="ff:ff:ff:ff:ff:ff", src=hw)/IP(src="0.0.0.0",dst="255.255.255.255") /
UDP(sport=68,dport=67)/BOOTP(op=1, xid=x_id, chaddr=hw_str)/DHCP(options=[("message-
type","request"),("requested_addr", offered_ip),("end")])


    #Sending the REQUEST for the offered IP address

    #Capturing the ACK from the server

    answr, unanswr = srp(dhcp_req_pkt, iface=pkt_inf, timeout = 2.5, verbose=0)


    #print answr

    #print unanswr

    #print answr[0][1][IP].src

    #print answr[0][1][BOOTP].yiaddr


    #The IP offered by the DHCP server to the client is extracted from the received answer

    offered_ip_ack = answr[0][1][BOOTP].yiaddr


    #DHCP Server IP/ID

    server_ip = answr[0][1][IP].src

    #print server_ip


    #Adding each leased IP to the list of leases

    all_given_leases.append(offered_ip_ack)


    #Adding the server IP to a list
```

```
    server_id.append(server_ip)

    client_mac.append(hw)

    return all_given_leases, server_id, client_mac


    ############## Application #3 - Part #3 ##############


################### DHCP RELEASE #################
def generate_dhcp_release(ip, hw, server):

    #Defining DHCP Transaction ID
    x_id = random.randrange(1, 1000000)
    hw_str = mac2str(hw)

    #Creating the RELEASE packet
    dhcp_rls_pkt = IP(src=ip,dst=server) / UDP(sport=68,dport=67)/BOOTP(chaddr=hw_str, ciaddr=ip,
xid=x_id)/DHCP(options=[("message-type","release"),("server_id", server),("end")])

    #Sending the RELEASE packet
    send(dhcp_rls_pkt, verbose=0)


    ############## Application #3 - Part #4 ##############
```

```
################## USER MENU #################

try:

   #Enter option for the first screen

   while True:

      print "\nUse this tool to:\ns - Simulate DHCP Clients\nr - Simulate DHCP Release\ne - Exit program\n"


      user_option_sim = raw_input("Enter your choice: ")


      if user_option_sim == "s":

         print "\nObtained leases will be exported to 'DHCP_Leases.txt'!"


         pkt_no = raw_input("\nNumber of DHCP clients to simulate: ")


         pkt_inf = raw_input("Interface on which to send packets: ")


         print "\nWaiting for clients to obtain IP addresses...\n"


         try:

            #Calling the function for the required number of times (pkt_no)

            for iterate in range(0, int(pkt_no)):

               all_leased_ips = generate_dhcp_seq()[0]


            #print all_leased_ips
```

```
            except IndexError:
                print "No DHCP Server detected or connection is broken."
                print "Check your network settings and try again.\n"
                sys.exit()

            #List of all leased IPs
            dhcp_leases = open("DHCP_Leases.txt", "w")

            #print all_leased_ips
            #print server_id
            #print client_mac

            #Print each leased IP to the file
            for index, each_ip in enumerate(all_leased_ips):

                print >>dhcp_leases, each_ip + "," + server_id[index] + "," + client_mac[index]

            dhcp_leases.close()

            continue

        elif user_option_sim == "r":
            while True:
                print "\ns - Release a single address\na - Release all addresses\ne - Exit to the previous screen\n"
```

```
user_option_release = raw_input("Enter your choice: ")


if user_option_release == "s":
    print "\n"


    user_option_address = raw_input("Enter IP address to release: ")


    #print all_leased_ips
    #print server_id
    #print client_mac


    try:
        #Check if required IP is in the list and run the release function for it
        if user_option_address in all_leased_ips:
            index = all_leased_ips.index(user_option_address)


            generate_dhcp_release(user_option_address, client_mac[index], server_id[index])


            print "\nSending RELEASE packet...\n"


        else:
            print "IP Address not in list.\n"
            continue
```

```
            except (NameError, IndexError):
                print "\nSimulating DHCP RELEASES cannot be done separately, without prior DHCP Client simulation."
                print "Restart the program and simulate DHCP Clients and RELEASES in the same program session.\n"
                sys.exit()

        elif user_option_release == "a":

            #print all_leased_ips
            #print server_id
            #print client_mac

            try:
                #Check if required IP is in the list and run the release function for it
                for user_option_address in all_leased_ips:

                    index = all_leased_ips.index(user_option_address)

                    generate_dhcp_release(user_option_address, client_mac[index], server_id[index])

            except (NameError, IndexError):
                print "\nSimulating DHCP RELEASES cannot be done separately, without prior DHCP Client simulation."
                print "Restart the program and simulate DHCP Clients and RELEASES in the same program session.\n"
                sys.exit()
```

```
                print "\nThe RELEASE packets have been sent.\n"

                #Erasing all leases from the file
                open("DHCP_Leases.txt", "w").close()

                print "File 'DHCP_Leases.txt' has been cleared."

                continue

            else:
                break

        else:
            print "Exiting... See ya...\n\n"
            sys.exit()

except KeyboardInterrupt:
    print "\n\nProgram aborted by user. Exiting...\n"
    sys.exit()

#End of program
```