Report

- This project is written in C using the LIBCAP and LIBNET libraries.

- The usage of this program is exactly as specified in the problem statement.

- The testing was done (for both INJECT and DETECT) using a normal Ubuntu system (running as host) and another Ubuntu system (running as guest victim on a Virtual Machine).

- A good portion of code related to sniffing packets was used from my previous Homework 2 project.

- Other standard codes available online were also used for various function. The references are present in the code as comments as well as a list mentioned below.

DNS Injection:

- The dns\_injector was started on my local machine, which was configured to listen on the default ethernet port (WIFI).

- LIBPCAP was used for sniffing. Only UDP packets on port 53 were considered. The rest were discarded. (It is also possible to specify this as a BPF filter, but for ease of use, it was used as default in the code.)

- Also, for testing, only DNS type A queries were considered.

- In case a file was specified, the injector would send an ip address according to the contents of the file. Otherwise, the current ip address of the local system was sent as fake.

- In the virtual machine, I used the “DIG” tool available on ubuntu for DNS queries.

- To find the difference in ip addresses returned, the dig command was run in two cases : one where the inject program was not running and the other when it was running.

- The inject program was run with two configurations: one without a file, and the other with a file with a specified mapping.

- For some cases, I ended up using PuntCat server instead of the google server at : (8.8.8.8). The slower servers show a better result and it becomes easy to spot a problem.

- Using a slower server results in my fake response getting received first.

- In the background, a tcpdump process is running that outputs it to a pcap file for use by the detect program.

(Some details of execution are mentioned with the images)

- In the inject program, if a file is not mentioned, I use my current ip address as the fake one and try and send it whenever there is a dns query.

- If there is a file mentioned, and a dns query for a website happens (which also has an entry in the mapping file), then the corresponding ip address according to the file is selected.

- After determining the ip, the libnet function calls are made for initialization, and creation of the required headers and payload. After the construction of the packet, it is sent via libnet\_write.

- Some of this code was used from sample programs of libnet already available as OpenSource.

- The required links are mentioned in the last page.

DNS Detection:

- Initial part of the code are the same as that of the inject program. Once it is determined that the packet is udp with port number 53, we proceed to check for duplicate entries.

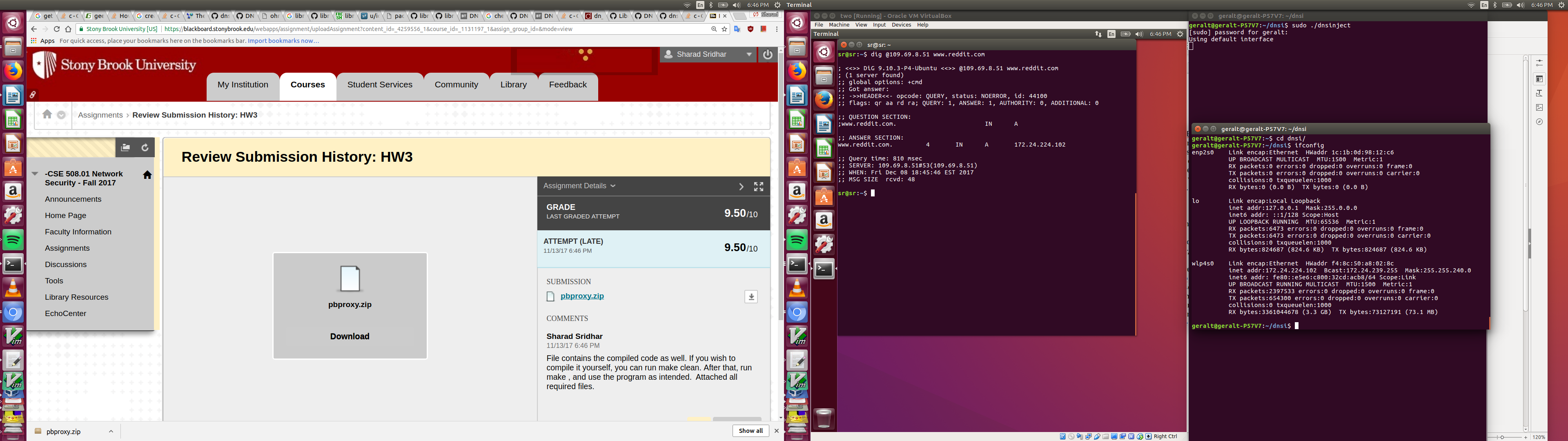
- With every packed examined, the dns response information is stored in a struct array.

- If there is any DNS response in the array with the same info (id and host name), we check for matching ip addresses in the answer array for matching dns responses. There can be multiple ips corresponding to the hostname due to Round Robin DNS. If there is no matching ip address, we can see that there is an attack attempt, as the victim receives multiple resolution for a domain name, and the required information is presented to the user.

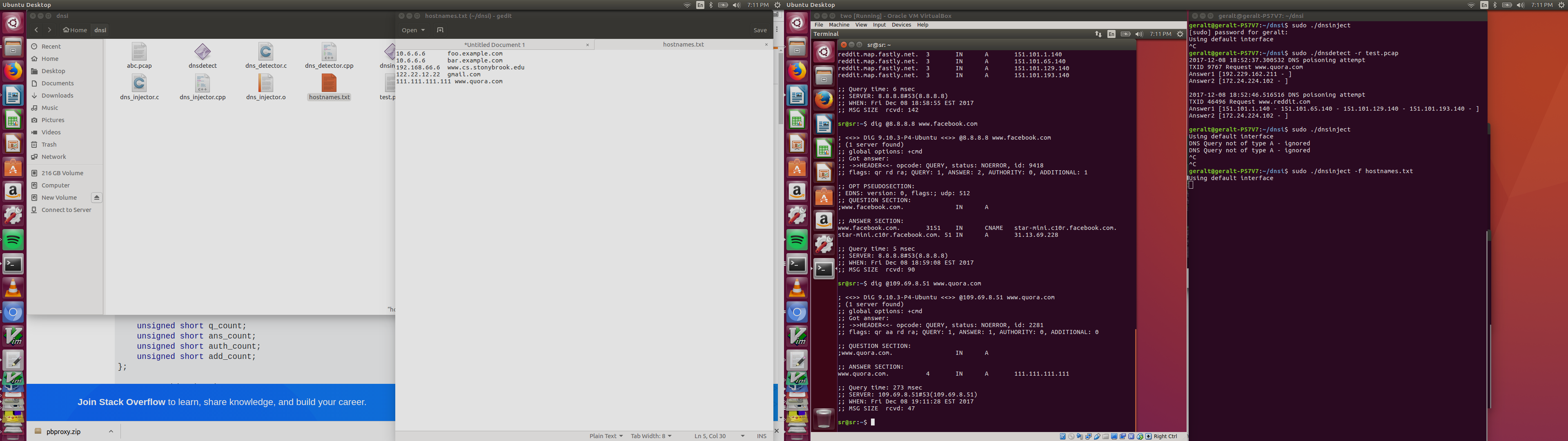
Case: Inject 1

The left terminal is on the VM. The right is my local machine.

The third terminal shows my current ip address which was sent as a fake reply to the victim.



Case : Inject 2



In this case, the file containing the mappings is used.

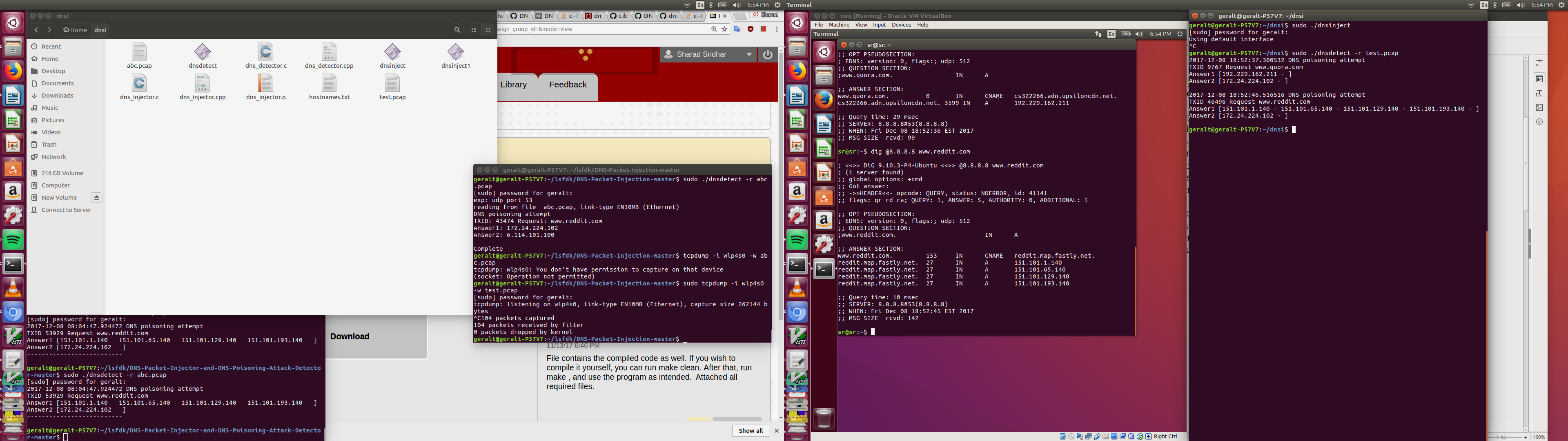
“[www.quora.com](http://www.quora.com/)” is mapped to “111.111.111.111”.

Using a slow server like Puntcat, I make sure my response is the earliest to arrive.

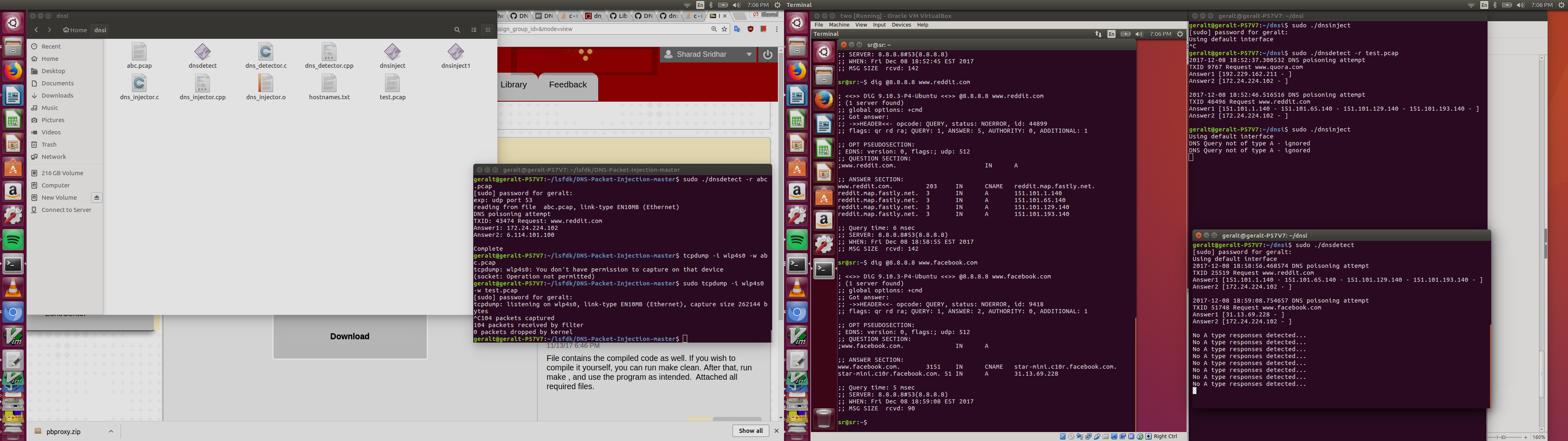
We can see that the dig command returns the fake ip provided by me.

Case : Detect 1

The figure shows when I ran the detect program on the pcap file I had created to monitor the traffic. There were attempts to poison the cache when I ran them through google’s servers (8.8.8.8). I had run the inject function without a file name, hence, my local ip address was sent as fake. This was detected correctly by the code.



Case : Detect 2



The left side vm terminal runs the dig command for “[www.reddit.com](http://www.reddit.com/)” and “[www.facebook.com](http://www.facebook.com/)”.

The right top terminal runs the inject program and tries to send my current ip address to the victim.

On another terminal, the detect program identifies the attack and prints the duplicate responses correctly.

Installation of required libraries can be done via:  
sudo apt-get install libnet-dev

sudo apt-get install libpcap-dev

To compile, just run the “make” command.

Running can be done as specified in problem statement. Please take a look at the images for examples.

References:

<http://www.binarytides.com/dns-query-code-in-c-with-linux-sockets/>

<https://stackoverflow.com/questions/2283494/get-ip-address-of-an-interface-on-linux>

<https://github.com/repolho/Libnet-1.1-tutorial-examples/blob/master/03_addr.c>

<https://github.com/maurotfilho/dns-spoof/blob/master/dns-spoof.c>

<https://github.com/sam-github/libnet/blob/master/libnet/sample/dns.c>

<https://gist.github.com/fffaraz/9d9170b57791c28ccda9255b48315168>