DoH-capable DNS forwarder

This project will require using Python to build a simple DNS forwarder with domain blocking and DoH capabilities. This DNS forwarder will need to do the following: (1) receive an arbitrary DNS message from a client, (2) check if the domain name should be blocked, and if so respond with an NXDomain message, (3) if the queried domain name is allowed, forward the DNS message to either standard DNS resolver or a DoH-capable resolver, (4) wait for the response from the resolver and forward it back to the client.

=== Details ===

Schema:

```
Client <==> DNS forwarder <==> DNS resolver < = = =
> (DNS NSes)
             ^^^^^^^^^^^^^
```

Command line parameters:

```
$ ./dns_forwarder.py -h
usage: dns_forwarder.py [-h] [-d DST_IP] -f
DENY_LIST_FILE
                        [-l LOG_FILE] [--doh] [--
doh_server DOH_SERVER]
optional arguments:
-h, --help   show this help message and exit
-d   DST_IP Destination DNS server IP
-f   DENY_LIST_FILE File containing domains to block
-l   LOG_FILE Append-only log file
--doh   Use default upstream DoH server
--doh_server DOH_SERVER  Use this upstream DoH
server
```

Requirements:

- If `--doh` or `--doh_server` are specified, the forwarder MUST forward the DNS query using the DoH protocol
- If `--doh` or `--doh_server` are not specified (in which case `-d` MUST be present), the forwarder MUST forward the DNS query using the DNS protocol

- The DNS forwarder MUST receive DNS messages from the client via a simple UDP server socket.
- When DoH is not used, the `-d` option will be specified and the forwarder must use a simple UDP client socket to forward the client's query to the DNS resolver
- The `DENY_LIST_FILE` file MUST contain a (potentially empty) list of domain names that MUST be blocked by the forwarder.

DoH REQUESTS

You are required to use GET requests as defined in RFC 8484. For instance, a DNS request for `(qname='example.com', qtype='A')` would look like:

```
https://8.8.8.8/dns-query?dns=AAABAAABAAAAAAAAB2V4YW1wbGUDY29tAAABAAE
```

Note: do not use the JSON API provided by some DNS operators. You can find more information here:

- https://developers.google.com/speed/public-dns/docs/secure-transports#doh
- https://developers.cloudflare.com/1.1.1.1/encrypted-dns/dns-over-https/make-api-requests/dns-wireformat
- https://datatracker.ietf.org/doc/html/rfc8484

NOTE: when working from the UGA campus network (including from your VM), use IP addresses (e.g., 1.1.1.1, 8.8.8.8, etc.) in the DoH query URL, because it appears that EITS is blocking some domain name strings related to DoH servers (e.g., dns.goole or cloudflare-dns.com).

DENY LIST FORMAT

The deny list provided in input will be a text file (not necessarily ending in .txt) containing one domain name per line. For instance:

```
www.example.com
cobweb.cs.uga.edu
yahoo.co.jp
```

only fully qualified domains should be blocked. For instance, in the above example only a domain matching yahoo.co.jp must be blocked (e.g., www.yahoo.co.jp should not be blocked).

Note: when a domain is blocked, your DNS forwarder MUST reply to the client with a properly formatted NXDOMAIN response.

LOG FILE ENTRY FORMAT

The log file should be a text file containing a record of all domain names and query types that have been requested, and whether the request was blocked or allowed. For instance:

```
www.google.com A ALLOW
google.com NS ALLOW
www.yahoo.co.jp A DENY
yahoo.co.jp MX DENY
www.youtube.com A ALLOW
www.example.com A DENY
...
```

SUBMISSION GUIDELINES

Create a directory named with your last name. Put your program's files under that directory. Then, create a Project3-FIRSTNAME-LASTNAME.tar.gz archive containing that directory. For instance, in my case those would be:

```
_ perdisci
  |___ dns_forwarder.py

Project3-ROBERTO-PERDISCI.tar.gz
```

Finally, submit the .tar.gz file through eLC.

Here is an example of how the code will be executed:

```
$ ./dns_forwarder.py --doh_server 1.1.1.1 -l
./queries.log -f ./deny_list.txt
```

Assuming `dns_fowarder.py` is running on `vmX.cs.uga.edu`, you could use `dig` to test it. For instance (this is only an example):

```
$ dig @vmX.cs.uga.edu -t AAAA www.example.com
```

and you should see a response similar to this one (again, this is only an example):

```
; <<>> DiG 9.10.6 <<>> @vmX.cs.uga.edu -t
AAAA www.example.com
; (1 server found)
;; global options: +cmd
```

```
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id:
49313
;; flags: qr rd ra ad; QUERY: 1, ANSWER: 1,
AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.example.com.        IN    AAAA
;; ANSWER SECTION:
www.example.com.    82730    IN    AAAA    2606:2800
:220:1:248:1893:25c8:1946
;; Query time: 10 msec
;; SERVER: vmX.cs.uga.edu#53(172.17.152.X)
;; WHEN: Thu Oct 14 16:01:17 EDT 2021
;; MSG SIZE  rcvd: 72
```

If www.example.com needs to be blocked (i.e., it's in the deny list), then the DNS forwarder should generate a response that looks like this (note: this is only an example):

```
; <<>> DiG 9.10.6 <<>> @vmX.cs.uga.edu -t
AAAA www.example.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id:
20003
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY:
0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.example.com.        IN    AAAA
;; Query time: 48 msec
;; SERVER: vmX.cs.uga.edu#53(172.17.152.X)
;; WHEN: Thu Oct 14 16:08:26 EDT 2021
;; MSG SIZE  rcvd: 47
```

Given that a few students are struggling with some parts of Project 3, here are some tips.

1) Basic examples on creating and parsing DNS messages with Scapy:

Example 1:

```
dns_req = DNS(rd=1, qd=DNSQR(qname='www.example.com'))
b = bytes(dns_req)
p = DNS(b)
p.show()
```

Example 2:

```
msg =
b'\x00\x00\x01\x00\x00\x01\x00\x00\x00\x00\x00\x00\x03abc\x07example\x03net\
x00\x00\x01\x00\x01'
DNS(msg).show()
```

2) Basic example on issuing HTTPS requests using the Requests library:

```
import requests
r = requests.get('https://1.1.1.1/')
r.status_code
```

You should get `r.status_code` = 200.
If the above works, then all the information you need to correctly make DoH requests can be found in these two links:

https://developers.cloudflare.com/1.1.1.1/encrypted-dns/dns-over-https/make-api-requests/dns-wireformat
https://docs.python-requests.org/en/master/user/quickstart/

NOTE: when working from the UGA campus network (including from your VM), use IP addresses (e.g., 1.1.1.1, 8.8.8.8, etc.) in the DoH query URL, because it appears that EITS is blocking some domain name strings related to DoH servers (e.g., dns.goole or cloudflare-dns.com).