Домашнее задание №3

Выполнил: Ланин Георгий Михайлович, БПИ202.

**Краткое описание:**

Программа написана языке Python, используется библиотека scapy версии 2.5.0. В проекте есть виртуальная среда. Чтобы ничего не устанавливать, нужно открыть терминал в папке проекта, ввести команду "./venv/Scripts/activate" для входа в среду, после чего ввести "python main.py". Если хотите использовать свой интерпретатор, то необходимо установить недостающие пакеты командой «pip install -r requirements.txt».

**Интерфейс программы:**

It is the program for scanning DNS packets and it supports commands:

1) 'capture {timeout}' - handles all packets and prints their body with timeout in seconds;

2) 'ipmx {Domain}' - find ip-addresses of mx address of domain

3) 'getip {Domain}' - find ip-addresses of domain;

4) 'exit' - stops program

**Инструкция:**

Для начала захвата пакетов DNS необходимо ввести команду «capture t», где t - это число секунд, в течение которых должен происходить захват.

Для определения IP-адресов определённого домена необходимо ввести команду «getip domain», где domain – это доменное имя, которое хочет проанализировать пользователь. Например, "getip yandex.ru".

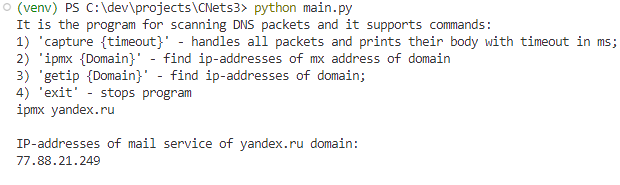
Для получения списка IP-адресов почтового сервиса определённого домена необходимо ввести команду «ipmx domain», где domain – это доменное имя, которое хочет проанализировать пользователь. Например, "ipmx yandex.ru".

Для выхода нужно дождаться возвращения в главное меню и ввести команду "exit" и нажать Ctrl+C.

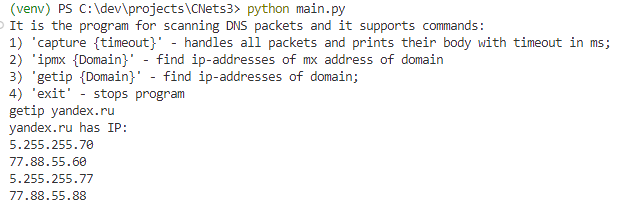
#Замечание №1: во время захвата пакетов Ctrl+C остановит функцию scapy, но не программу, не советую использовать данный метод остановки вне главного меню программы.

#Замечание №2: константа CONST\_SERVER хранит в себе IP-адрес, к которому обращается программа (по умолчанию 8.8.8.8), при желании его можно заменить.

**Пример работы ipmx**



**Пример работы getip**

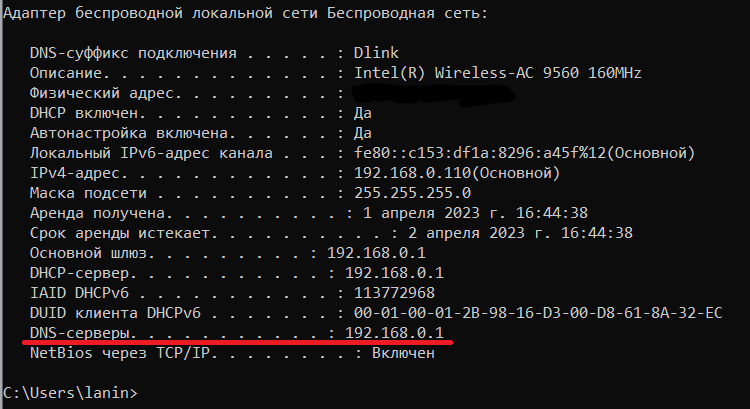


**Работа с корневыми серверами.**

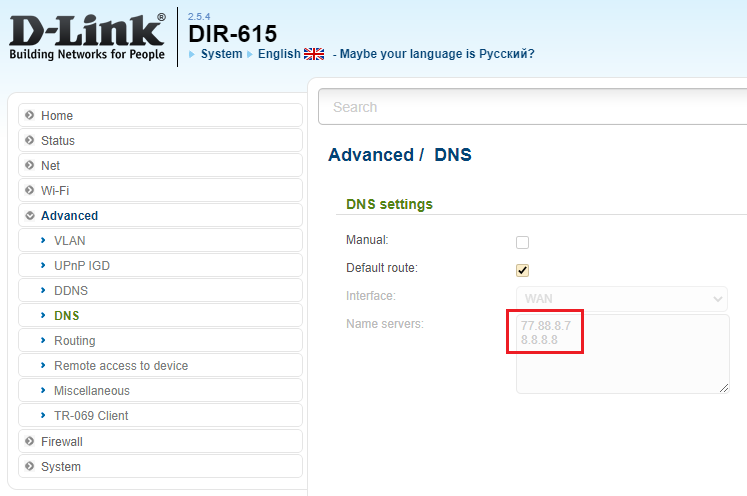
1) Выберем корневой серверj.root-servers.net с адресом192.58.128.30

2) Теперь определим, к каким DNS-серверам обращается наше оборудование:

ipconfig/all

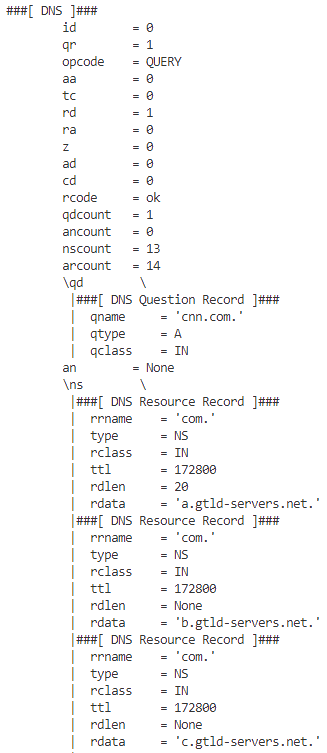


Идём в настройки роутера:



Согласно выкладке whois адрес 77.88.8.7 закреплён за сервером от Яндекс, а адрес 8.8.8.8 за сервером от Google. Любопытно, что по умолчанию роутер получает адреса dns серверов от провайдера, а тот, в свою очередь, прописал именно адрес 77.88.8.7, хотя у Яндекса есть ещё 77.88.8.8 и 77.88.8.88. Возможно, это объяснимо тем, что используемая мной сеть предоставляется общежитием, ей требуется фильтрация трафика, которую Яндекс предоставляет для этого [сервера](https://dns.yandex.ru/)

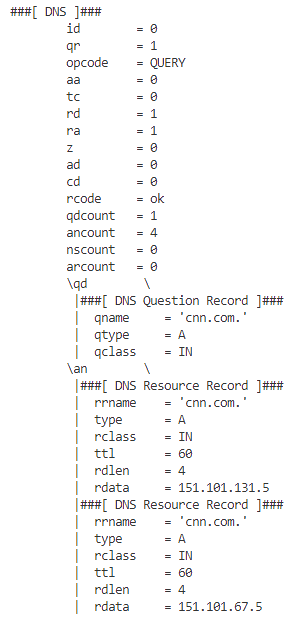
Задав корневой сервер и выполнив команду, я получил пустой ответ программы, посмотрел содержимое result.show(), где корневой сервер справедливо ответил, что не знает IP-адрес нашего домена, но знает IP-адреса серверов, ответственных за домен первого уровня. Вот как это выглядит:



aa=0 – сервер не authoritative для этого запроса

ra=0 – рекурсия недоступна для этого запроса

Проверим, как нам отвечает сервер от Google:

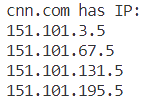


aa=0 – сервер не authoritative для этого запроса

ra=1 – рекурсия доступна для этого запроса

Сервер вернул 4 ресурсные записи, а моя программа, соответственно, вывела 4 IP-адреса. В scapy бит rd (рекурсия желательна) по умолчанию 1, поэтому в случае, когда сервер поддерживает рекурсивный запрос, мы можем получить ответ.

Если бы мы начали опрашивать сервера, которые корневой сервер выдал для домена .com, мы нашли бы authoritative сервер для cnn. Моя программа не делает это автоматически, поэтому приведу ответы сервера от Google, который предоставляет мне провайдер.







На всякий случай покажу ниже, что отвечал корневой сервер для hse.ru и draw.io (для cnn скриншоты были выше):

|  |  |
| --- | --- |
|  |  |

Опять же, сервер вернул IP-адреса серверов, ответственных за домен первого уровня для каждого доменного имени.

**Пример работы capture**

Команда: "capture 10"

Открыт: сайт <https://edition.cnn.com/videos/politics/2023/02/12/mike-rounds-social-security-sotu-vpx.cnn> Просто тыкал по cnn, чтобы нахватать запросов.  
  


Для перехвата пакетов использовался фильтр filter='dst port 53'

После первого запуска выяснилось, что библиотека читает все пакеты, идущие на этот порт, и не все из них про DNS. Многие добавляют к данному фильтру ограничение на UDP, так как чаще всего DNS использует UDP-пакеты, однако согласно документации [Microsoft](https://learn.microsoft.com/ru-ru/troubleshoot/windows-server/networking/dns-works-on-tcp-and-udp) может быть использован и TCP. Поэтому я фильтрую пакеты вручную, проверяя, содержат ли они «слой» DNS.

Здесь мы видим, что компьютер обратился к роутеру, запросив информацию о некоторых ресурсах с сайта. Меня смутило, что все пакеты представлены в виде Standart query и среди них нет Standart Query Response. Wireshark помог понять, что такие пакеты могут приходить к нам на другие порты, поэтому фильтр был изменён на dst port 53 or src port 53

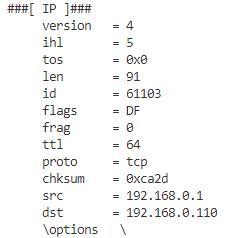
Это сработало, вот пример сводки:

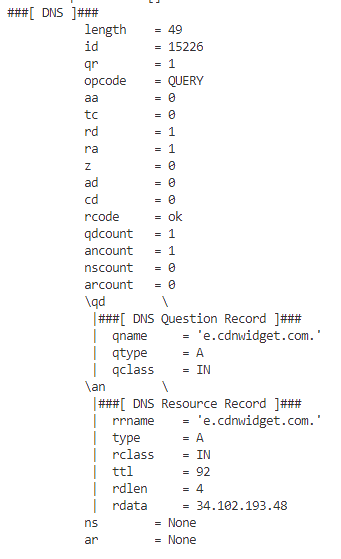


Здесь и TCP, и UDP, и DNS Query, и DNS Answer.

Вот несколько примеров пакетов:

Кадр представлен частично (без Ethernet)

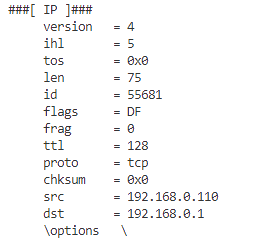


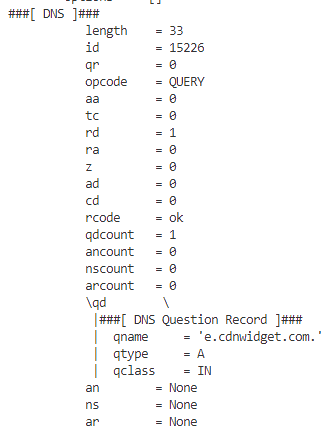


Роутер отправил компьютеру пакет.

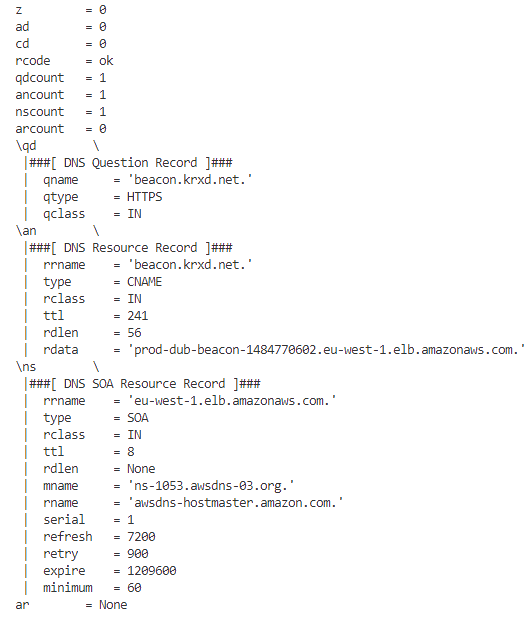
Здесь мы получили response (qr=1), сервер не authoritative, ответ поместился в один пакет(TC=0), рекурсия желательна и доступна, ошибок не было, в разделе вопроса была только одна запись, запрашивался IPv4-адрес сайта e.cdnwidget.com. в сети интернет, в ответ была получена ресурсная запись с IPv4-адресом сайта, кэшировать стоит результат стоит в течение 92 секунд.

А вот так выглядел запрос компьютера за 6 тысячных секунды до:





А вот другой пакет:



Здесь компьютер запрашивал запись типа «HTTPS» для доменного имени beacon.krxd.net., на что был получен ответ с записью типа CNAME с Alias, а также запись типа SOA в разделе NS с указанием зоны, где хранится эталонная запись.