Une image contenant flou, ciel nocturne

Description générée automatiquement

Networks Architectures project

Server project (DNS / DHCP trace)

Becker lucas, Ngo Thomas, Bue Mathieu, cordier Benoît

IOS 1 – Promo 2022

## description

Les fonctionnalités demandées pour le serveur sont :

Server under (Windows / Linux):

1) Connection to your DNS / DHCP log server (TCP)

2) Possibility to log in several @MACs managed by SQLite

3) Retrieval of DNS and DHCP requests in an SQLite database by @MAC

4) Possibility to display the logs by @MAC or by the OUI file (Manufacturer)

5) Possibility of sorting logs by (@IP, Date, Time ...)

6) Managing a list of unauthorized DNS

7) Notification if a DNS is in the list of unauthorized DNS

8) Notification if a DHCP @IP is in the list of unauthorized @MACs

9) Save logs to one file per day

10) Your original feature

Bonus:

1) Ability to read and send an alert email if for example DDOS attempt

## Fonctionnement

Blablabla

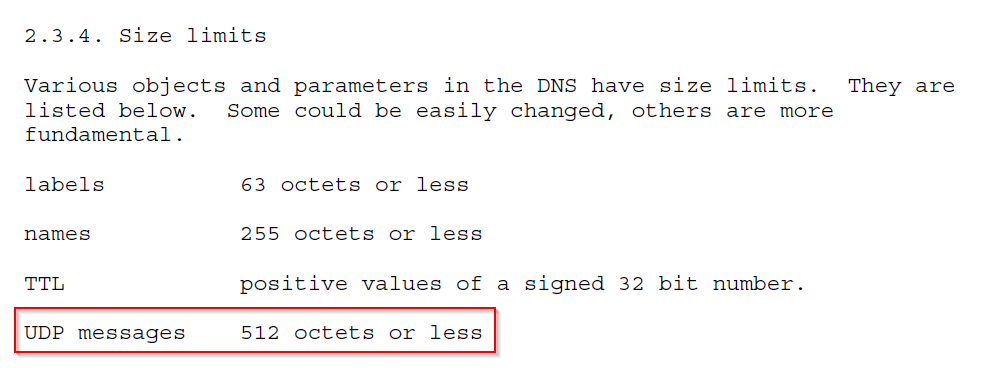
## 10) Notre caracteristique originale

Après échanges avec M. Livolsi nous nous sommes particulièrement intéressés aux serveurs DNS. Il a évoqué l’existence du DNS over HTTPS et du DNSSEC qui sont des évolutions plus sécurisées du DNS de base. Il est vrai que les informations transmises dans les paquets DNS sont sensibles, il parait ainsi logique de chercher à les sécuriser. Cependant la quasi-totalité des routeurs et servers DNS grand public n’implémentent pas ces sécurités.

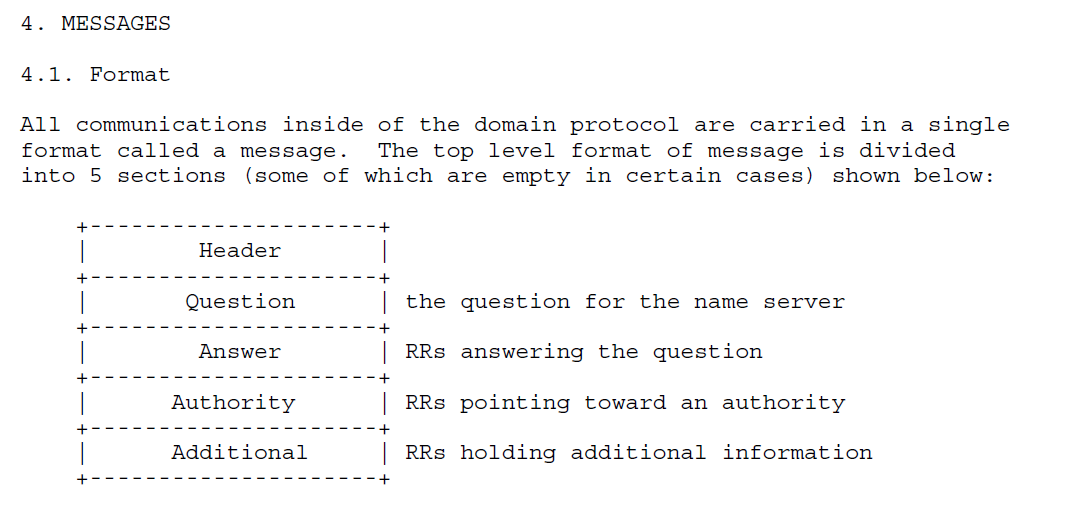
Nous avons donc décider de construire un vrai serveur DNS basique en Python pour se familiariser plus en profondeur avec les DNS.

Ce serveur communique avec des clients DNS (comme les navigateurs ou la commande **DIG**) via les sockets.

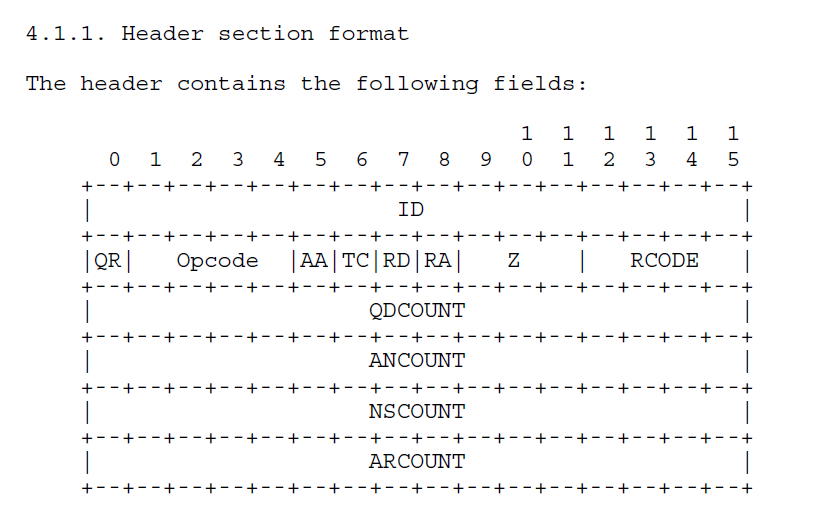
Pour commencer il faut comprendre la structure d’un paquet DNS. Pour ce faire, il faut lire la documentation des RFC 2929 et 1035.



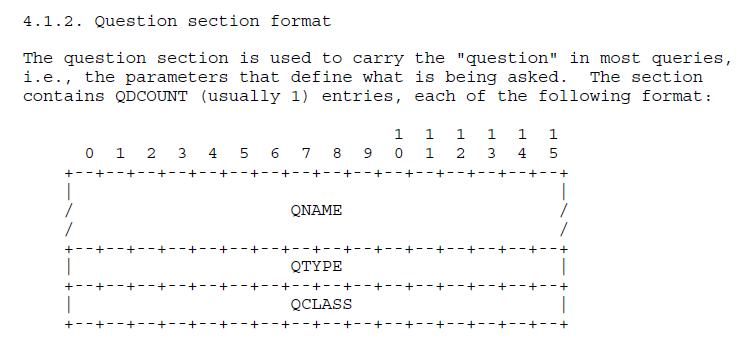
Notre serveur est assez basique et se contente de récupérer une Query et d’envoyer une Answer. Les tailles des paquets sont relativement petites, on utilise donc le protocole UDP.



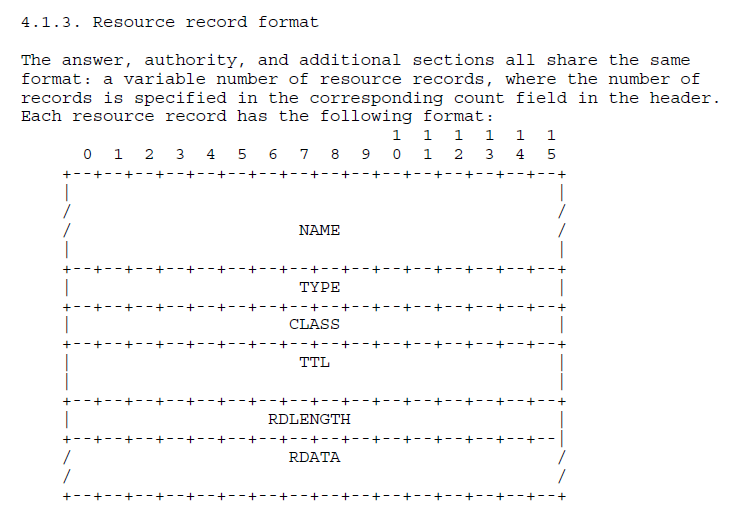
Les Query DNS se composent des sections **Header** et **Question**. Pour les Answer DNS, il faut ajouter à cela la section **Answer**. Nous n’utilisons pas les autres sections.



Seul l’ID change entre les requêtes DNS dans notre utilisation. Ce qui nous permet d’entrer tous les autres champs en dur lors de la construction du paquet de réponse.

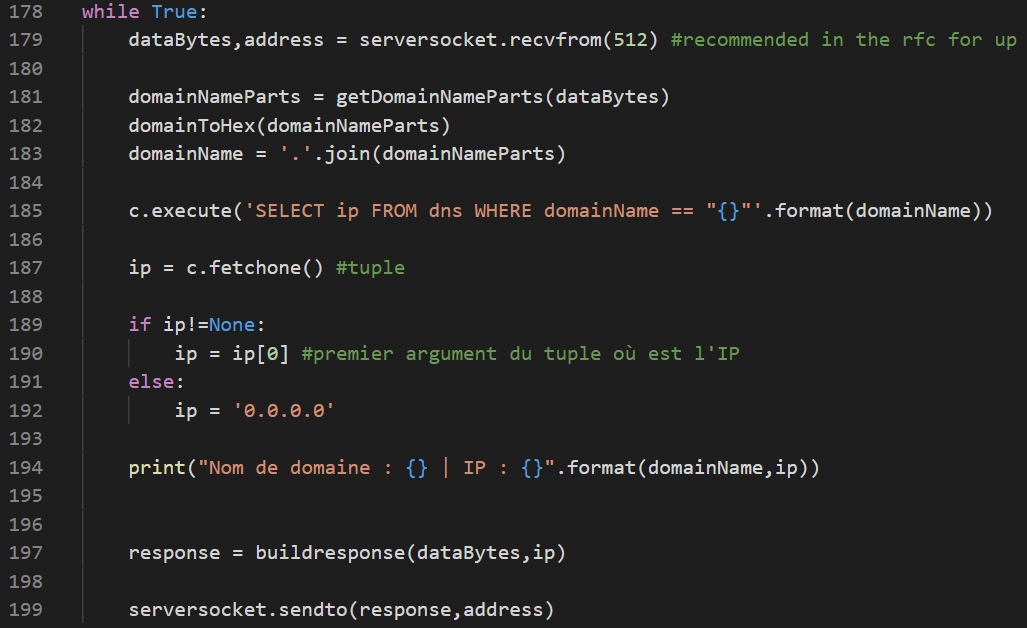


Ici c’est seulement le nom de domaine (QNAME) que l’on doit modifier entre les paquets.

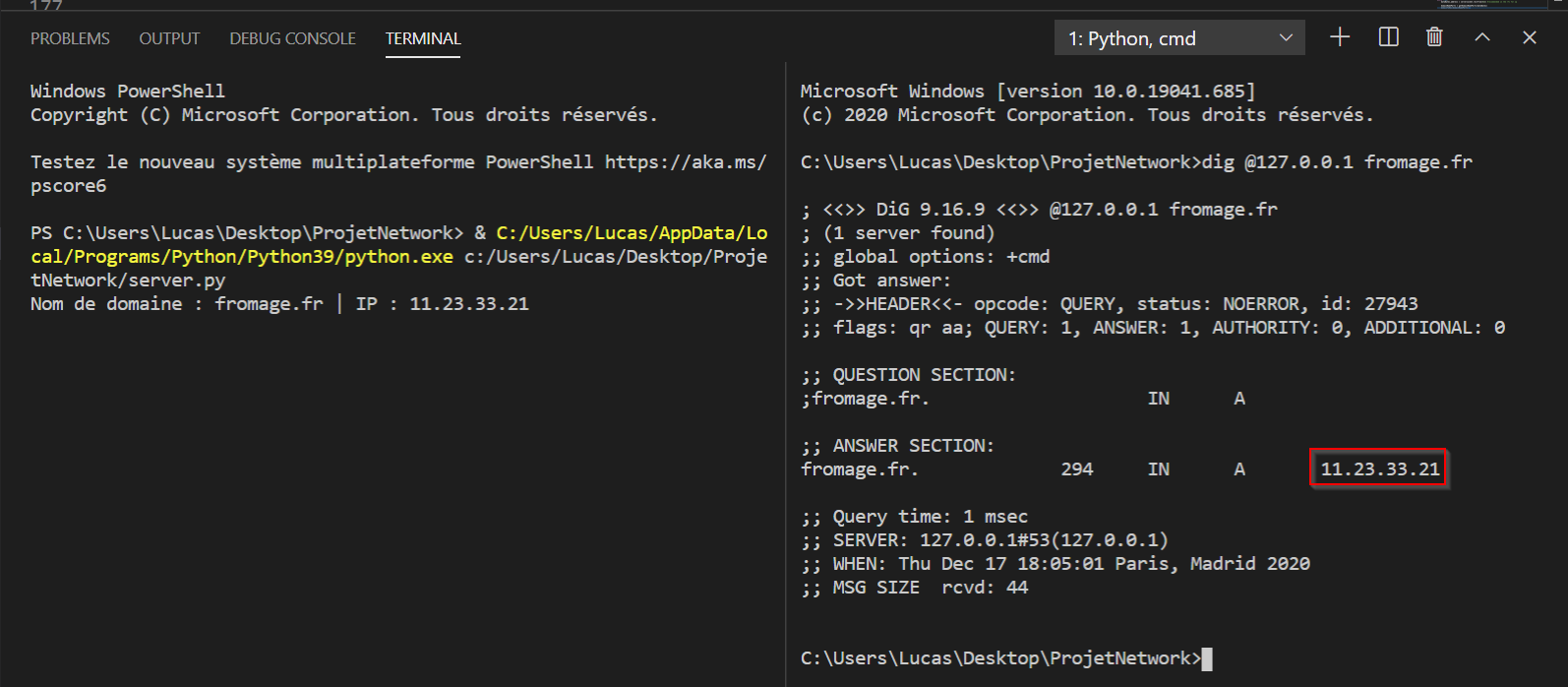


Après avoir renseigné tous ces champs, le paquet de réponse est prêt à être envoyer.

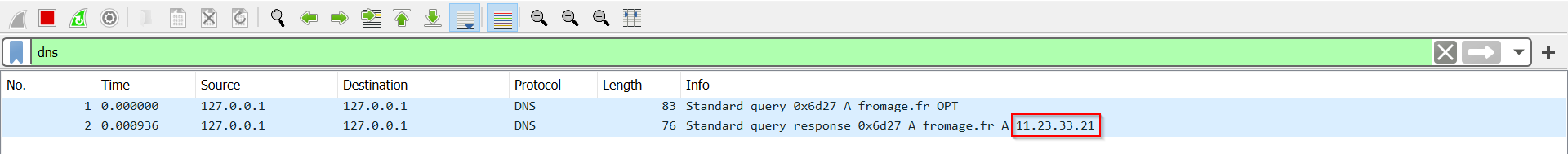
Voici la boucle principale du programme :



Le serveur cherche dans sa base de données s’il possède l’IP du nom de domaine dans la Query. S’il ne possède pas l’IP, il renvoie 0.0.0.0. Sinon il renvoie un paquet de réponse avec l’IP.



On voit ici que le client **DIG** reçoit un paquet DNS en bonne et due forme en tant que réponse.



On peut vérifier sur Wireshark que l’échange s’est bien déroulé.

Grâce à cette caractéristique originale du projet global, nous avons pu approfondir notre connaissance des serveurs DNS. Maintenant que nous comprenons en détails comment les échanges se déroulent, nous pouvons maintenant nous intéresser aux évolutions plus sécurisées telles que DNS over HTTPS ou DNSSEC.