🔄 Introduction – Le DNS dynamique expliqué simplement

Dans une infrastructure réseau vivante, où les machines peuvent apparaître, disparaître ou changer d'IP (comme avec des clients DHCP), le DNS classique ne suffit pas. On a besoin que les **enregistrements DNS puissent se mettre à jour automatiquement**, en temps réel, sans intervention manuelle.

C'est exactement ce que permet le **DNS dynamique** (*Dynamic DNS*, ou DDNS).

🎯 À quoi ça sert, concrètement ?

Le DNS dynamique permet à un serveur DHCP (comme OPNsense) ou à une machine autorisée d'envoyer automatiquement une requête au serveur DNS pour :

- Ajouter un enregistrement A ou PTR
- Modifier ou supprimer une entrée existante
- Faire tout ça **de façon sécurisée**

Ce mécanisme est indispensable dès qu'on veut :

- Associer un **nom à une machine DHCP** dans une zone DNS
- Gérer dynamiquement des **clients mobiles, serveurs temporaires, etc.**
- Éviter les mises à jour manuelles dans les fichiers de zone

---

🔐 Comment ça fonctionne : l'authentification via TSIG

Pour éviter que n'importe qui ne vienne modifier la zone DNS, on utilise une **authentification par clé privée partagée** : c'est le protocole **TSIG** (*Transaction SIGnature*).

Voici le principe :

1. Une **clé secrète est générée** et connue uniquement du serveur DNS (BIND) et du client autorisé (DHCP ou `nsupdate`)
2. À chaque mise à jour DNS, le client **signe la requête avec cette clé**
3. Le serveur vérifie la signature :
    - ✅ si la clé et la signature sont valides → il accepte la mise à jour
    - ❌ sinon → rejet

Le tout repose sur des **algorithmes HMAC** (SHA256, SHA512…) et une horodatation pour éviter les attaques par rejeu.

🧠 En résumé :

Le DNS dynamique permet à des clients ou des serveurs DHCP de **mettre à jour automatiquement et en toute sécurité les enregistrements DNS**, sans toucher manuellement aux fichiers de zone.

C'est **rapide**, **fiable**, **sécurisé par TSIG**, et **indispensable dans un environnement réseau dynamique**.

Mon serveur DNS : 192.168.10.122

Ma zone : sadek.local

Fichier de la zone :

```
  GNU nano 5.4                                                    db.sadek.local
$TTL 86400
sadek.local.  in SOA ns.sadek.local root.local(
                1       36000 3600 360000 86400)
                IN NS ns.sadek.local.

ns.sadek.local.         IN      A       192.168.10.122
```

Il faut d'abord créer la clef TSIG

tsig-keygen -a HMAC-SHA256 dyn-key > /etc/bind/tsig.key

```
root@ossec-bastion:/etc/bind# tsig-keygen -a HMAC-SHA256 dyn-key > /etc/bind/tsig.key
root@ossec-bastion:/etc/bind# cat tsig.key
key "dyn-key" {
        algorithm hmac-sha256;
        secret "3ZhRJmYCfQ09qTEgKEUrp4/nHCiTOYtvZiLOg3wUnv8=";
};
root@ossec-bastion:/etc/bind#
```

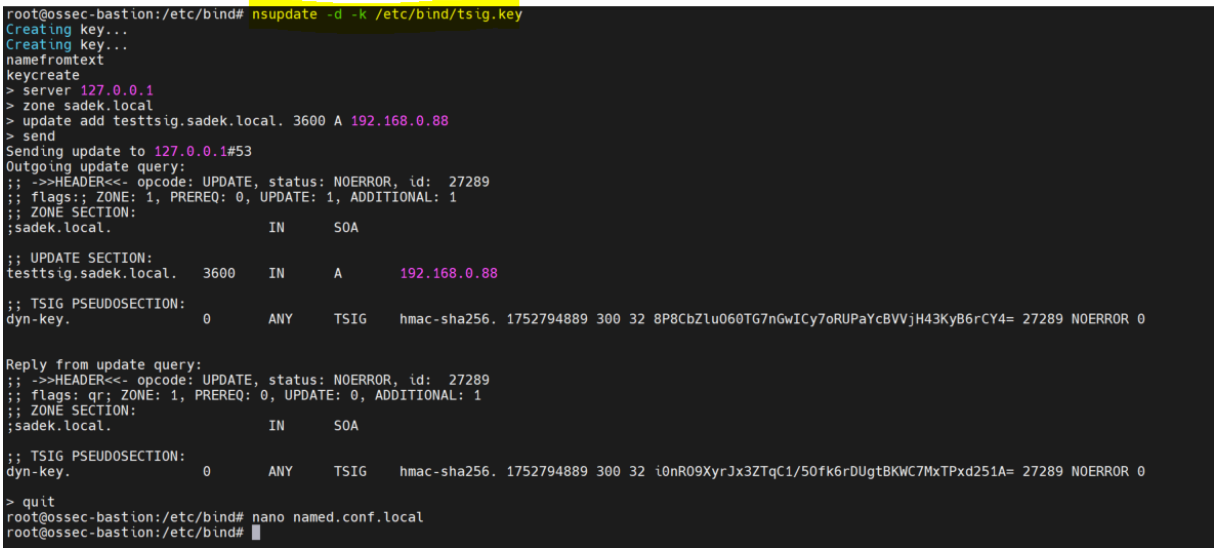
Ensuite il faut que je l'ajoute dans named.conf.local comme ci-dessous

```
  GNU nano 5.4                                                    named.conf.local
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "sadek.local" {
                type master;
                file "/etc/bind/db.sadek.local";
                allow-update { key "dyn-key"; };
};
```

Nous avons déjà créer la zone maintenant nous allonw faire un test avec la commande

Apres plusieurs soucis il faut absolument rester sur cette commande et ne pas utiliser la DNSSEC

La commande nsupdate est aussi a faire correctement en utilisant l'argument k et spécifier chemin de la clef + d pour debug



Et si on fait un test avec dig on voit bien que l'enregistrement existe maintenant

Chaque clef = un fichier different on ne peut pas mettre plusieurs clefs dans le fichier

Les modifications sont ecrit dans un fichier « .jnl » qui est en vérité un fichier binaire qu'ilk ne faut jamais modifier à la main

```
root@ossec-bastion:/var/cache/bind# dig @127.0.0.1 testtsig.sadek.local

; <<>> DiG 9.16.50-Debian <<>> @127.0.0.1 testtsig.sadek.local
; (1 server found)
;; global options: +cmd
;; Got answer:
;; WARNING: .local is reserved for Multicast DNS
;; You are currently testing what happens when an mDNS query is leaked to DNS
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58052
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: f1d8e9acf45052fa010000006879888a99c2935488a82bca (good)
;; QUESTION SECTION:
;testtsig.sadek.local.          IN      A

;; ANSWER SECTION:
testtsig.sadek.local.   3600    IN      A       192.168.0.88

;; Query time: 0 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Fri Jul 18 01:34:34 CEST 2025
;; MSG SIZE  rcvd: 93

root@ossec-bastion:/var/cache/bind#
```

Mis en place sur opnsense sur mon routeuru voici ce que je vois