Sécurisation du DNS

1. Limitation des clients

Il est possible de limiter les requêtes autorisées. La directive **allow-query** dans le fichier de configuration permet de définir les hôtes ou réseaux auxquels un serveur acceptera de répondre.

Limitation des clients autorisés dans le fichier named.conf

```
allow-query { réseaux-autorisés; };
```

Où *réseaux-autorisés* représente la ou les adresses de réseaux ou d'hôte qui pourront s'adresser au serveur.

2. Utilisation d'un compte de service

a. Pourquoi un compte de service ?

Aux origines, il était fréquent de faire tourner un serveur **bind** sous l'identité du compte d'administration **root**. C'est-à-dire que le compte **root** était propriétaire du processus. Les conséquences pouvaient être fâcheuses : si du code sensible (dangereux) était envoyé au processeur par l'exécutable **named**, il l'était au nom de **root**, c'est-à-dire avec les pleins pouvoirs sur le système. Or, cette situation présente des risques. Il peut s'agir de bogues contenus dans le code exécutable de **named**, ou bien de vulnérabilités du programme qui permettraient à une personne mal intentionnée d'envoyer du code exécutable via le processus.

La solution retenue est en général d'exécuter **named** sous une identité différente de **root**, et d'utiliser un compte de service : un compte utilisateur ne permettant pas de connexion directe au système, mais qui sera propriétaire du processus. Ainsi, si du code malicieux venait à être exécuté via le processus **named**, il n'aurait pas plus de pouvoir que ceux du compte de service, et ne pourrait donc pas mettre en péril le système.

La plupart des implémentations modernes de **bind** exploitent nativement l'utilisation d'un compte de service.

Compte de service named sur une distribution Red Hat

Le compte est créé automatiquement à l'installation du service.

```
[root@RH9 root]# grep named /etc/passwd
named:x:25:25:Named:/var/named:/sbin/nologin
```

b. Lancement de named avec un compte de service

Encore une fois, toutes les implémentations de **bind** utilisées dans des distributions modernes de Linux exploitent nativement un compte de service. La démarche indiquée ici est donc nativement incluse dans les scripts de lancement de service.

Syntaxe simplifiée de la commande named pour utilisation d'un compte de service

```
named -u utilisateur
```

Éléments	
named	L'exécutable principal de bind. Dans la plupart des implémentations, lancé depuis le script de gestion du service.
utilisateur	Appelé par le paramètre "-u", indique le compte de service propriétaire du processus. Ce compte doit naturellement être défini dans le fichier /etc/passwd.

3. Bind en mode chroot

a. Pourquoi enfermer le processus ?

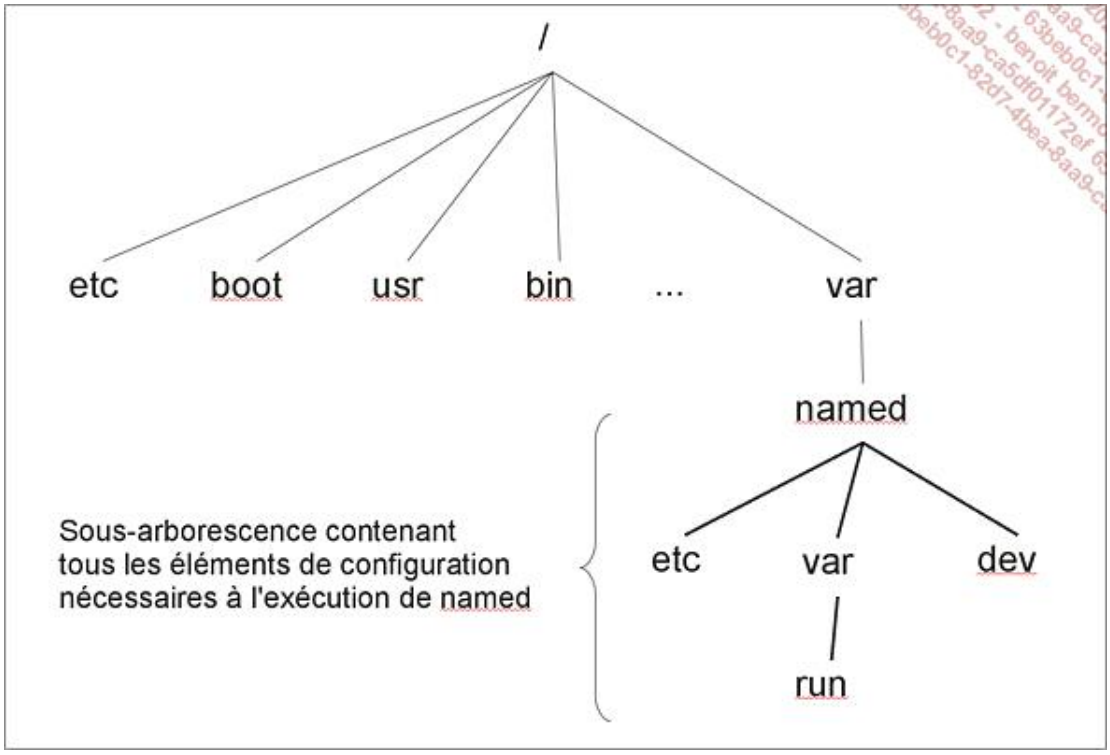
Nous avons vu plus haut qu'une utilisation malencontreuse du processus **named** pouvait entraîner le risque d'actions dangereuses pour le système. L'enfermement du processus dans un répertoire dédié permet de limiter les risques. Le but est de faire croire au processus qu'il s'exécute dans un système ordinaire, alors qu'il est cantonné dans une arborescence parallèle, et qu'il ne peut en aucun cas interagir avec le reste du système. Le terme « enfermement » n'est pas usurpé, on parle en anglais de le mettre « in jail », c'est-à-dire en prison.

On dit alors qu'on utilise **bind** en mode **chroot**, contraction de **change root** (changement de racine).

Il est recommandé d'utiliser le mode **chroot** avec un compte de service. Un processus qui aurait les prérogatives de **root** pourrait s'octroyer le droit de sortir du répertoire où il est enfermé.

b. Création de l'environnement nécessaire

Dans la mesure où le processus est berné et qu'il croit s'exécuter dans un environnement ordinaire, il doit avoir à sa disposition tous les éléments nécessaires à son fonctionnement. Il faut bien comprendre que le processus n'aura aucun moyen d'aller chercher quoique ce soit en dehors de son répertoire. La mise en place d'un **bind** en mode **chroot** suppose donc une phase préliminaire de création de son environnement de travail.



Étapes de création de l'environnement de travail :

- Création du répertoire de chroot.
- Création de la fausse arborescence "/" dans le répertoire de chroot. Tous les répertoires utilisés par le processus named doivent s'y trouver.
- Copie des fichiers de configuration dans le répertoire de chroot.
- Lancement du processus en mode chroot.

c. Lancement du programme en mode chroot

Il n'est pas vraiment difficile de lancer **bind** en mode **chroot**.

Syntaxe de la commande named pour utilisation en mode chroot

```
named -c config -u utilisateur -t repertoire
```

Éléments employés dans la syntaxe :	
named	L'exécutable principal de bind. Dans la plupart des implémentations, lancé depuis le script de gestion du service.
config	Facultatif. Indique le fichier de configuration à employer au chargement. En principe /etc/named.conf ou /etc/bind/named.conf.
utilisateur	Le compte de service propriétaire du processus. Ce compte doit naturellement être défini dans le fichier /etc/passwd.
repertoire	Le répertoire dans lequel named sera enfermé. Souvent /var/named.

Il sera bon de vérifier dans les journaux que le processus parvient bien à démarrer dans son nouvel environnement. En général, quelques essais sont nécessaires.

4. Échange sécurisé entre serveurs

De nombreuses fonctions d'Internet reposent sur le DNS, depuis la simple navigation jusqu'à l'envoi d'e-mails. Les tentatives de phishing, de plus en plus nombreuses, montrent bien les dangers que peut présenter une incertitude sur la résolution de noms. Si quelqu'un se connecte au site de sa banque en ligne en utilisant l'url exacte mais que le nom est résolu en l'adresse IP d'un faussaire, les conséquences peuvent évidemment être dramatiques. Dans le cas du DNS, la sécurisation reposera surtout sur l'authentification et l'intégrité des données. C'est-à-dire qu'on veut être certain que c'est bien le bon serveur qui nous répond, et que les données ne subissent pas de modification pendant le trajet.

Nous allons utiliser ici le mécanisme **TSIG : Transaction SIGnature** (*signature des transactions*). Ce mécanisme repose sur la présence d'un secret partagé par les serveurs qui échangent des données.

a. Génération du secret partagé

Il existe un outil de génération des clés : **dnssec-keygen**. Il a de nombreux usages, mais nous le verrons ici dans le cadre d'une exploitation TSIG.

Syntaxe dnssec-keygen pour utilisation dans le cadre de TSIG

```
dnssec-keygen -a HMAC-MD5 -b taillecle -n nametype nomsecret
```

dnssec-keygen : variables et paramètres	
-a HMAC-MD5	-a définit la méthode de cryptage. HMAC-MD5 est la seule valeur supportée pour TSIG.
-b *taillecle*	-b définit la valeur de la clé employée. Pour HMAC-MD5, *taillecle* doit être compris entre 1 et 512. 128 est une valeur courante généralement satisfaisante.
-n *nametype*	-n définit le propriétaire de la clé. Dans le cadre d'un fonctionnement TSIG, *nametype* aura généralement la valeur HOST pour signifier que la sécurisation se passe de machine à machine.
nomsecret	Le nom du secret. Peut être n'importe quelle chaîne alphanumérique.

La commande aboutit à la génération de deux fichiers **Knomsecret.+xxx.yyyyy.key** et **Knomsecret.+xxx.yyyyy.private**.

Exemple d'utilisation de dnssec-keygen

```
donald:~# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST supersecret
Ksupersecret.+157+26824

donald:~# cat Ksupersecret.+157+26824.key
supersecret. IN KEY 512 3 157
  yItYGlAQtGcM7VqGjZdJAg==

donald:~# cat Ksupersecret.+157+26824.private
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: yItYGlAQtGcM7VqGjZdJAg==
```

b. Déclaration du secret dans named.conf

Sur les serveurs concernés par la sécurisation, on inclura dans le fichier **named.conf** la définition du secret.

Syntaxe de la déclaration de secret dans named.conf

```
key nomsecret {
   algorithm hmac-md5;
   secret "yItYGlAQtGcM7VqGjZdJAg==";
};
```

Éléments employés dans cette syntaxe :	
key	Annonce la déclaration de la clé.
nomsecret	Le nom du secret utilisé à la génération.
algorithm	A pour paramètre le type d'algorithme utilisé.
hmac-md5	Obligatoire pour TSIG.
secret	A pour paramètre le secret généré dans le fichier Knomsecret.+xxx.yyyyy.key (la chaîne de caractères entre guillemets).

c. Les deux serveurs doivent utiliser la clé

La clé partagée étant déclarée sur les deux serveurs, il faut maintenant leur faire savoir qu'ils doivent l'utiliser pour garantir la sécurité de certaines communications. Il faudra donc ajouter une nouvelle commande dans **named.conf**.

Syntaxe d'utilisation de la clé dans named.conf

```
server ip_dest {
   keys { nomsecret; };
};
```

Éléments employés dans cette syntaxe :	
server	Annonce un comportement pour un serveur déterminé.
ip_dest	L'adresse IP du serveur pour lequel directive s'applique.
keys	Annonce le secret utilisé pour sécuriser les échanges.
nomsecret	Le nom du secret utilisé à la génération.

d. Tout service est refusé en l’absence de signature

Le paragraphe précédent a donné aux serveurs la capacité de communiquer de façon sécurisée. Il faut ensuite rendre obligatoire cette sécurisation pour les requêtes entre serveurs dans le cadre d’une délégation par exemple.

Syntaxe

```
zone « truc.fr » {
    type master;
    file « db.truc.fr »;
    allow-recursion { key supersecret; };
};
```