Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE
PREMIER MINISTRE

**S . G . D . S . N**
*Agence nationale de la sécurité des systèmes d'information*
*CERTA*

Paris, le 25 juillet 2008
N° CERTA-2008-INF-002

**Affaire suivie par :**
CERTA

# NOTE D'INFORMATION DU CERTA

## Objet : Du bon usage du DNS



## Gestion du document

| | |
|---|---|
| Référence | CERTA-2008-INF-002 |
| Titre | Du bon usage du DNS |
| Date de la première version | 25 juillet 2008 |
| Date de la dernière version | – |
| Source(s) | |
| Pièce(s) jointe(s) | Aucune |

TAB. 1 – *Gestion du document*

Une gestion de version détaillée se trouve à la fin de ce document.

## Table des matières



Secrétariat général de la défense et de la sécurité nationale – ANSSI – COSSI – CERTA
51, bd de La Tour-Maubourg **Tél.**: 01 71 75 84 50 **Web**: http://www.certa.ssi.gouv.fr
75700 Paris 07 SP **Fax**: 01 71 75 84 70 **Mél** : certa-svp@certa.ssi.gouv.fr

# 1 Introduction

Le DNS au sens large fait partie des sujets de préoccupation récurrents et bien réels, car il s'agit d'un système central sur lequel reposent actuellement beaucoup d'autres technologies. Le cloisonnement et l'indépendance de certaines couches protocolaires restent plus théoriques qu'autre chose. Le DNS est *de facto* un élément sensible d'une architecture réseau. Par ailleurs, le système est également lié à une hiérarchie mondiale qui oblige des échanges et des accès auprès de celle-ci. Malgré sa popularité, le DNS n'est pas un protocole trivial. Plusieurs évolutions ont complexifié son comportement global, le rendant plus obscur aux administrateurs des réseaux.

Ce document n'a absolument pas la prétention de présenter en quelques pages l'ensemble du système. En revanche, il a pour objectif de présenter quelques risques qui peuvent concerner l'utilisateur ou l'administrateur d'un réseau. Il apporte donc quelques éléments de réponses à ces risques, en s'appuyant sur des incidents déjà constatés ou pouvant se réaliser de manière relativement réaliste.

Le document est structuré de la manière suivante : une première partie revient sur le DNS et son organisation, en particulier sur quelques points de fonctionnement qu'il est nécessaire de bien assimiler. Une seconde partie trace plusieurs risques actuels pesant sur cette architecture. Une troisième partie présente quelques bonnes pratiques déduites des précédents risques cités.

Ce document se destine avant tout aux personnes curieuses de comprendre les enjeux de DNS, mais possédant également quelques expériences d'informatique et du monde des réseaux.

# 2 Retour sur le DNS

## 2.1 Rapide historique

Le DNS ("*Domain Name System*" ou *système d'adressage par domaines* selon le JO « Vocabulaire de l'informatique et de l'Internet » du 16/03/1999) est un service d'annuaire. Tout comme un annuaire téléphonique permet de déterminer un numéro de téléphone à partir d'un nom, l'interrogation DNS fournit l'adresse IP 213.56.176.2 pour joindre le site « www.certa.ssi.gouv.fr » sur l'Internet (adresse utilisée à la création de cette note).

Il s'agit d'un procédé courant en informatique qui consiste à dissocier le nom d'une ressource de sa localisation physique (l'adresse IP en première approximation). L'adresse de la ressource est alors déterminée en interrogeant un annuaire. Outre le fait qu'il est plus « facile » pour un être humain de retenir « www.certa.ssi.gouv.fr » que 213.56.176.2, ceci permet au gestionnaire du site de modifier son adresse IP (changement de fournisseur d'accès, par exemple) en ayant juste à mettre à jour le serveur DNS plutôt que d'avoir à informer tous les clients potentiels de la nouvelle adresse IP.

Puisque c'est une étape indispensable pour l'accès à une majorité de ressources, le DNS est un service critique pour le fonctionnement d'Internet.

## 2.2 Hiérarchie et différents acteurs

Le DNS est organisé sous la forme d'une base de données répartie et hiérarchisée.

Les données associées aux noms sont aussi appelées des « ressources ». Chaque niveau hiérarchique est soit autoritaire pour une ressource, soit délègue cette autorité à un niveau inférieur.

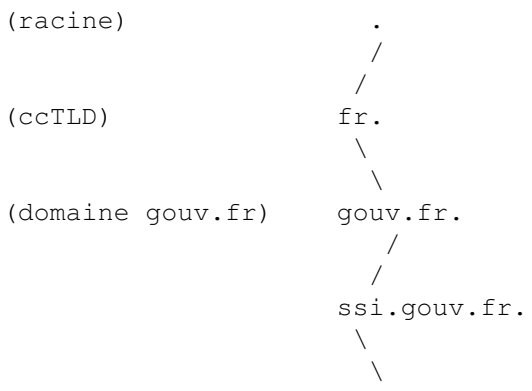
Outre la racine de la hiérarchie qui se présente par un point « . », il existe les domaines de plus haut niveau, aussi connus sous l'acronyme TLDs (*Top Level Domains*). Les plus populaires sont `.COM`, `.ORG`, `.NET`, `.FR`, etc.

Une liste des TLDs valide est maintenue à l'adresse :

http://data.iana.org/TLD/tlds-alpha-by-domain.txt

Les serveurs responsables de la racine de la hiérarchie sont appelés serveurs racines (il en existe 13 à la date de rédaction de ce document).

On distingue généralement parmi les TLDs ceux attribués en fonction des pays[1] comme `.FR`, les ccTLDs (*country code TLDs*), et ceux plus génériques, les gTLDs, comme `.COM`, `.NET` ou `.ORG`. Le schéma ci-dessous présente un exemple de hiérarchie :




[1] Les codes des pays sont définis par l'ISO 3166.

```
certa.ssi.gouv.fr.
```

Il est important de comprendre que toute autorité parente a le pouvoir de définir la valeur d'une ressource, et que, par conséquent, tout gestionnaire d'un domaine fait implicitement confiance aux gestionnaires des domaines parents. La base de données des serveurs de noms est constituée « d'enregistrements de ressources », ou *Ressource Records* (RR). Ces enregistrements sont répartis en classes. La classe la plus fréquente est la classe Internet ou « IN ».

Quel que soit le niveau de hiérarchie considéré, les échanges se font sur Internet par des requêtes utilisant les protocoles de transport (UDP et/ou TCP selon les cas) et les serveurs sont en écoute sur le port standard 53.

## 2.3 Partage de responsabilités

La hiérarchie des noms de domaine est découpée en « zones ». Chaque zone correspond à une responsabilité administrative et possède au moins un serveur d'autorité. Ce dernier contient la base complète de la zone (il retourne des réponses d'autorité, ou AA, pour *Authoritative Answer*). Bien souvent, chaque zone possède plusieurs serveurs, on discerne alors :

- le serveur primaire (maître) ;
- les serveurs secondaires (esclaves).

Les serveurs secondaires interrogent régulièrement le serveur d'autorité primaire pour savoir si les données ont changé depuis la dernière mise à jour. Ils se « synchronisent » avec le primaire en effectuant régulièrement des `transferts de zone` (cf. section 2.7).

Ils utilisent un numéro (conventionnellement construit à partir de la date et de la version), toujours incrémenté et identifiant la version d'une zone. Ils comparent donc pour la mise à jour le champ SERIAL d'un RR particulier (nommé SOA) de la zone transmis par le serveur primaire qu'ils connaissent. Si ce numéro a augmenté, ils chargent les nouvelles données.

Parmi les enregistrements existants, les principaux sont :

- l'enregistrement SOA (*Start Of Authority*) qui spécifie le serveur de noms qui est la meilleure source d'information pour les données d'une zone, l'adresse électronique d'un contact technique et des paramètres d'expiration. Par exemple :

```
$dig @dns.ssi.gouv.fr ssi.gouv.fr SOA
:: ANSWER SECTION:
ssi.gouv.fr. 7200 IN SOA dns.ssi.gouv.fr resptech.ssi.gouv.fr  2008010101
                                                    3600 1800 604800 3600
```

- l'enregistrement NS précisant les serveurs de noms faisant autorité sur la zone ;
- l'enregistrement d'alias et d'adresse (CNAME et A) précisant les correspondances entre noms -> adresses ("A"), ou alias -> noms ("CNAME", cf. section 4.7.1) ;
- l'enregistrement PTR qui pointe quelque part dans l'espace de nommage. Il est utilisé à l'inverse pour les correspondances entre adresses -> noms (l'adresse est alors représentée en sens inverse puis le suffixe `in-addr.arpa`).
- l'enregistrement MX (*Mail eXchange*) indique l'adresse d'un ou plusieurs serveurs de gestion du courrier adressé à un utilisateur dans ce domaine.

```
La correspondance entre nom et adresse :
$dig www.certa.ssi.gouv.fr

;; ANSWER SECTION:
www.certa.ssi.gouv.fr.  5729   IN    A  213.56.176.2

La correspondance inverse :
$dig PTR 2.176.56.213.in-addr.arpa
ou
$dig -x 213.56.176.2
```

```
;; ANSWER SECTION:
2.176.56.213.in-addr.arpa. 159200 IN CNAME 2.0-29.176.56.213.in-addr.arpa.
2.0-29.176.56.213.in-addr.arpa. IN  21600 PTR  www.certa.ssi.gouv.fr.
```

L'alias utilisé (partie gauche de l'enregistrement CNAME) utilise ici une propriété définie dans le RFC 2317. Il permet de déléguer une partie du IN-ADDR.ARPA dans le cas des plages d'adresses utilisant des préfixes plus grands que 24 bits.

Un serveur de noms charge les informations d'une zone. Il ne contient pas nécessairement toutes les informations d'un domaine si celui-ci a délégué d'autres zones à d'autres serveurs de noms. Une zone se délimite par les délégations, car elle n'intègre pas les données déléguées. Le serveur délégant contient seulement des pointeurs vers d'autres serveurs de noms qui font autorité sur les sous-domaines.

## 2.4 Serveurs racines

Les serveurs racines sont responsables de la racine de la hiérarchie DNS (« . »). Ce sont eux qui répondent pour le haut de la hiérarchie DNS. Ces derniers sont inscrits par défaut dans tout serveur DNS pour pouvoir répondre à toute requête. Ils se trouvent dans des fichiers dédiés selon les systèmes d'exploitation et doivent correspondre à la liste définie à l'adresse :

http://www.internic.net/zones/named.root

Les serveurs racines sont actuellement au nombre de 13. Ils sont localisés à divers endroits du globe (cf. section 2.10) et sont nommés en utilisant les 13 premières lettres de l'alphabet.

`{A-M}-root-servers.net`

Leur contenu permet de rediriger vers les serveurs gérant chaque TLD. Il est résumé à l'adresse suivante :
http://www.internic.net/zones/root.zone

Une quantité importante de requêtes est envoyée en permanence à ces serveurs. À valeur d'illustration, voici quelques chiffres récupérés suite à une initiative nommée DITL 2008 : des chercheurs ont analysé pendant une journée (le 19 mars 2008) le trafic de 8 des 13 serveurs racines. 7,56 milliards de requêtes ont été comptabilisées, dont 700 000 environ en TCP, depuis 5,6 millions d'adresses IP distinctes. Cela correspond à une moyenne de 10 000 requêtes par seconde et par serveur. 1,38% de ces requêtes concernent en réalité des plages d'adresses privées précisées par le RFC 1918 (ces adresses ne doivent en principe pas être visibles de l'Internet public). De nombreux clients interrogent directement ces serveurs racines. Les chercheurs constatent également un peu plus de 25% des requêtes qui s'adressent à des TLD invalides comme « local », « localhost », « domain », « invalid », « wpad », etc.

Les serveurs racines sont sous la responsabilité de l'ICANN mais gérés par des consortiums délégués. Un comité consultatif nommé RSSAC (*DNS Root Server System Advisory Committee*) a pour mission de prévenir l'ICANN de toute intervention sur les serveurs racines et de fournir tout conseil sur les besoins opérationnels de ceux-ci (machines physiques, systèmes d'exploitation, logiciels, connectivités réseau, environnement, etc.).

- Présentation du RSSAC par l'ICANN :
  http://www.icann.org/committees/dns-root/
- "Operational Criteria For Root Name Servers", RFC 2010, Octobre 1996 :
  http://www.ietf.org/rfc/rfc2010.txt
- "Root Name Server Operational Requirements", RFC 2870, Juin 2000 :
  http://www.ietf.org/rfc/rfc2870.txt

L'IANA (*Internet Assigned Numbers Authority*) produit un fichier particulier donnant les caractéristiques des serveurs racines, le `root zone file`, qui est stocké sur quelques serveurs. Chaque opérateur des serveurs racines se procure ensuite le fichier dans ceux-ci et le distribue sur les serveurs qu'il gère de la manière qu'il le souhaite.

Les règles de diffusion ainsi que le contenu du fichier sont détaillées par l'IANA à la page :
http://www.iana.org/domains/root/

## 2.5 Récursivité vs. itérativité

Le DNS étant une architecture distribuée, il y a deux approches pour un serveur quant à la façon de répondre aux requêtes :

- récursive : un premier serveur ne possédant pas la réponse à la requête qui lui est soumise prend la requête à son compte, détermine un second serveur compétent et transmet la requête. Il continue ensuite de contacter d'autres serveurs en fonction des réponse retournées, jusqu'à ce qui'il puisse apporter la réponse complète ;

– itérative : un premier serveur ne possédant pas la réponse fournit au client la meilleure réponse qu'il connaisse. Le client effectue alors lui-même de nouvelles requêtes pour résoudre.

Dans le mode récursif, le serveur prend tout le processus de résolution à sa charge pour apporter une réponse définitive. Dans le mode itératif, le serveur ne fait qu'orienter le client vers d'autres serveurs adaptés.

Une interrogation DNS type implique de manière générale quelques acteurs ayant des rôles différents :

– un client aussi appelé « résolveur » privilégiant une interrogation récursive vers un ou plusieurs serveurs caches (configurés manuellement ou spécifiés dans un bail DHCP. . . ) et d'interpréter leurs réponses. Il n'a pas l'« intelligence » suffisante pour suivre les différentes réponses et laisse cette tâche aux serveurs de noms locaux ou précisés dans la configuration réseau. Ces derniers feront alors les opérations nécessaires pour fournir une réponse finale ;
– un serveur de noms interrogé, obligé de répondre complètement à la requête envoyée par le client. Dans le cas contraire, il doit envoyer une réponse précisant que la donnée requise ou le nom demandé n'existe pas. Ce serveur ne fait pas nécessairement autorité sur les informations demandées et pose donc les questions à d'autres serveurs (en mode itératif ou récursif). Dans le cas le plus complet, il demandera d'abord aux serveurs racine, puisque par définition chaque serveur de noms connaît les noms et les adresses des serveurs racines.
– un serveur de cache DNS qui est souvent commun au serveur récursif et qui prend en charge de fournir des réponses complètes. Il a une bonne vue de l'ensemble des requêtes et donc des optimisations possibles ;
– d'autres serveurs DNS, dont un serveur DNS autorité pour la ressource demandée dont la réponse sera transmise au client par le serveur récursif précédent.

Le dessin 1 reprend le processus complet sous forme de schéma d'une résolution. Les étapes sont les suivantes :

1° le serveur de noms local Serv1 reçoit une requête récursive en provenance d'un poste client pour résoudre test.gouv.fr.
2° le serveur envoie une requête à l'un des serveurs racines ;
3° le serveur racine lui indique les serveurs de noms de `fr`. L'un d'eux est Serv2. Il ne le contacte pas directement car il est en mode itératif. Il laisse cette tâche à Serv1 ;
4° Serv1 continue le processus de résolution car il est en mode récursif et interroge donc Serv2 pour résoudre test.gouv.fr ;
5° Serv2 renseigne Serv1 sur d'autres serveurs dont Serv3. Il ne le contacte pas directement car il est en mode itératif ;
6° Serv1 envoie une requête à Serv3 pour résoudre test.gouv.fr ;
7° Serv3 répond avec autorité. Il est également en mode itératif ;
8° Serv1 transfère la réponse au poste client.

Il s'agit bien sûr d'un processus simplifié. Les serveurs intermédiaires interrogés peuvent eux-même être en mode récursif et prendre en charge une partie des requêtes. Ce processus ne tient pas compte non plus d'éventuelles mises en cache.

Le choix de ces modes de configuration est primordial et a de très fortes implications dans l'architecture DNS. De manière générale, les serveurs racines ainsi que les serveurs publics doivent être en mode itératif, tandis que des serveurs au cœur des réseaux d'opérateurs ou permettant de résoudre un nom depuis un Intranet doivent offrir le service récursif pour les utilisateurs clients ou internes uniquement.

Un serveur est dit « récursif ouvert » s'il effectue des tâches récursives pour tout client de l'Internet. Des tests effectués sur un échantillon de serveurs de noms publics montre en janvier 2008 que plus de la moitié d'entre eux sont ouverts. Des articles concernant ces problèmes de configuration sont disponibles aux adresses suivantes :

– D. Wessels, "Strange Things Found in an Open Resolver Survey", WIDE/CAIDA Workshop, janvier 2008 : http://www.caida.org/whorshops/wide/0801/slides/dw-openresolvers.pdf
– D. Wessels, "Finding Open DNS Resolvers", DNSOPS 2006 : http://public.dns-oarc.net/files/dnsops-2006/Wessels-Openresolvers.pdf
– J. Kristoff, "Probing for Open DNS Resolvers", Midwest Security Workshop, septembre 2006 : http://condor.depaul.edu/ jkristof/slides/msw2-dnsprobing.pdf
http://condor.depaul.edu/ jkristof/orns/

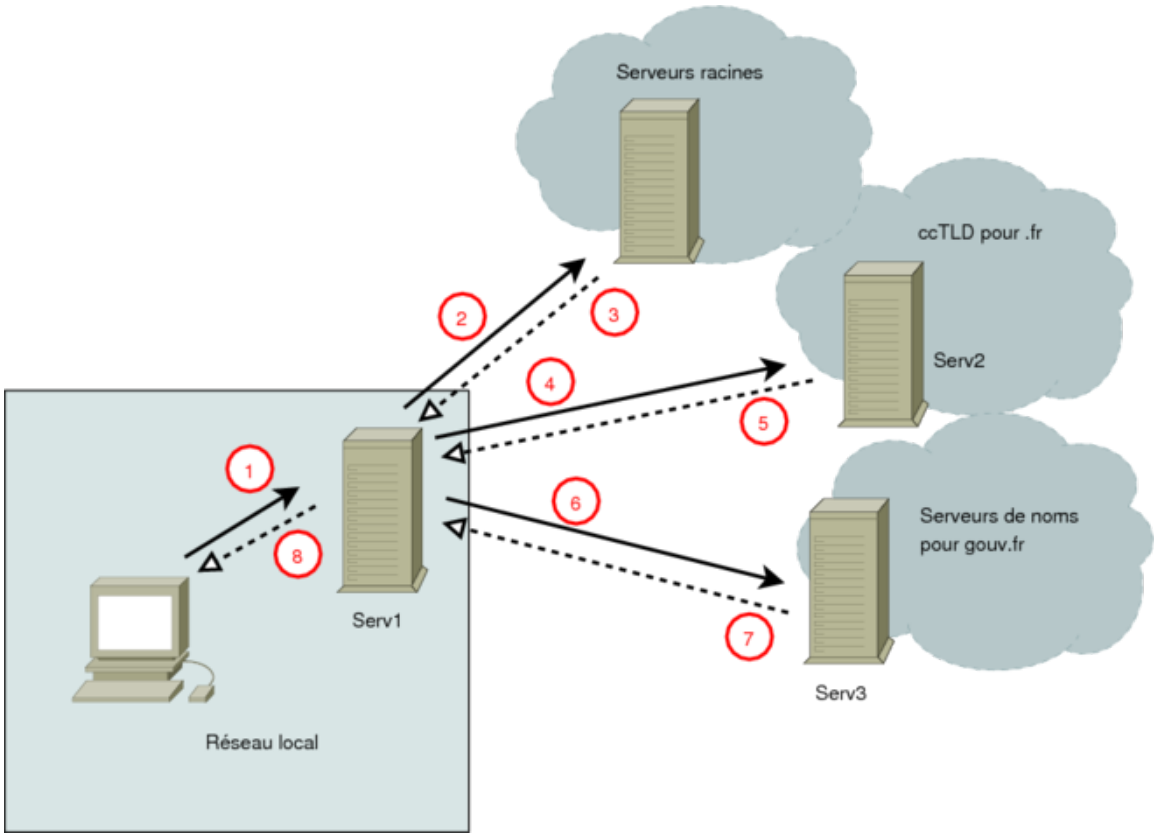



FIG. 1: Schéma simplifié des communications pour une résolution de noms

## 2.6 Mise en cache

### 2.6.1 Pour les serveurs

Il existe deux types de services DNS distincts :

- les serveurs qui sont autorités pour une ou plusieurs zones données ;
- les serveurs cache qui relaient des requêtes de plusieurs clients et une mise en cache des réponses dans le but de diminuer à la fois le trafic DNS nécessaire et les ressources affectées à la résolution pour chaque client.

Malheureusement, ces deux fonctions sont trop souvent confondues et mutualisées sur le même serveur, phénomène accentué par la confusion qu'introduit l'interrogation en mode récursif. Une minimisation du risque lié à ces services conduit à séparer ces deux fonctions et, selon l'architecture réseau, implique souvent des localisations différentes au sein du réseau.

Il faut comprendre qu'un serveur peut mettre en cache le résultat de l'information demandée, y compris les réponses négatives où le serveur d'autorité signale que le nom ou le type de données recherchées n'existe pas. C'est le principe du « cache négatif » (RFC 2308).

Le serveur répondant par des données depuis son cache (donc dont la durée de vie, *Time to Live* ou TTL, spécifiée par le serveur autoritaire n'a pas encore expiré) signale qu'il n'a pas autorité sur la zone demandée.

### 2.6.2 Pour les postes clients

Le client possède également un cache, qu'il est possible d'interroger et d'effacer par des commandes propres au système d'exploitation utilisé. Celles-ci ne sont pas toujours très documentées :

- Sous Microsoft Windows :
  - `net stop dnscache` pour arrêter le service responsable de la mise en cache ;
  - `ipconfig /displaydns` pour afficher le contenu du cache DNS ;
  - `ipconfig /flushdns` pour effacer le contenu du cache DNS ;
  - le registre
    `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters`

pour modifier le temps maximal (en secondes) qu'une réponse DNS doit rester en cache. Il suffit d'ajouter ou modifier les variables `MaxCacheTtl` et `MaxNegativeCacheTtl`. Les valeurs (quand les variables existent) sont par défaut de 86400s (1 jour) pour les réponses abouties et 900s (15min) pour les réponses négatives.

  - Base de connaissances Microsoft, "Comment faire pour désactiver la mise en cache DNS côté client dans Windows XP et Windows Server 2003 :
    http://support.microsoft.com/kb/318803/fr

- Sous Linux :
  - les entrées DNS ne sont pas mises naturellement en cache ;
  - si le service `nscd` est utilisé, `/etc/rc.d/init.d/nscd restart` pour effacer le contenu du cache. En fonction des systèmes, `rndc flush` (utilitaire BIND) peut également fonctionner. La section 4.7.8 donne quelques exemples d'applications existantes.
- Sous Mac OS :
  - `lookupd -flushcache` pour effacer le contenu du cache sous MacOS 10.4.x ;
  - `lookupd -configuration` permet d'affiner certaines propriétés comme la mise en cache (clé booléenne `ValidateCache`, nombre maximal d'entrées dans le cache `CacheCapacity` ou délai d'existence dans le cache `TimeToLive`, temps maximum d'attente de réponse du serveur `Timeout`, etc.) ;
  - `dscacheutil -flushcache` sous MacOS 10.5.x pour effacer le contenu du cache.

Les serveurs utilisés pour faire du cache perdent en général leur contenu après un simple redémarrage du service. Il ne faut pas oublier également que certaines applications peuvent aussi décider de mettre en cache. C'est le cas de certains navigateurs comme `Mozilla Firefox`. On peut trouver dans leurs configurations respectives des options. Dans Firefox, il faut accéder à l'option `network.dnsCacheEntries` dans le menu `about:config`.

Après une certaine période, le résolveur rejette l'enregistrement du cache. Cette période est indiquée dans le TTL associé à l'enregistrement de ressource DNS.

## 2.7 Transferts de zones

La section 2.3 a mentionné que plusieurs serveurs se partagent la responsabilité des zones. Les informations concernant ces zones doivent restées cohérentes parmi les serveurs et impliquent donc un processus de mise à jour. Les mises à jour des zones peuvent se faire de manière manuelle en modifiant pour chaque serveur les fichiers adéquats. Il existe aussi une possibilité de mise à jour dynamique qui s'appuie sur un numéro de série des SOA. Celui-ci s'incrémente à chaque modification de fichier afin que les serveurs secondaires récupèrent les nouvelles données. Il est codé sur 32 bits.

En principe, les serveurs secondaires effectuent régulièrement un test pour savoir s'il faut lancer un transfert de zone. L'intervalle entre ces tests est la « période de rafraîchissement ». Cependant, cette durée peut être relativement longue et pendant celle-ci, le serveur secondaire n'est pas à jour. Il existe donc un autre mécanisme par lequel le serveur primaire informe les serveurs secondaires de toute modification de zone. Il est détaillé dans le standard RFC 1996 : `NOTIFY`. La trame est semblable à la réponse d'une requête d'enregistrement SOA de zone : elle inclut l'enregistrement SOA de la zone ayant subi des modifications. En recevant un tel paquet, les serveurs secondaires écourtent alors leur période de rafraîchissement, répondent à ce paquet et lancent le processus de transfert de zone. Le serveur n'émet pas directement de message NOTIFY instantanément, dès qu'une modification a été effectuée. Il temporise en général et attend quelques minutes ou un nombre donné de modifications.

En pratique, les zones peuvent contenir un nombre très important de noms (cas de machines configurées en DHCP et avec Active Directory). Dans ce cas, il n'est pas réaliste d'effectuer des transferts de zone complets, sous peine de surcharger la bande passante disponible et l'activité des serveurs. Il existe donc la possibilité d'effectuer des transferts incrémentaux : IXFR. Le serveur secondaire précise de quelle version de zone il dispose et le serveur maître ne lui envoie ainsi que les modifications apportées à cette version. Si le transfert de zone est complet, on parle alors de requête AXFR. La méthode IXFR n'est applicable que pour des modifications de zone dynamiques où le serveur conserve les traces des modifications.

## 2.8 Notions de maîtres/esclaves et serveurs cachés

Il existe une terminologie parlant de relation maître/esclave entre serveurs qui peut prêter à confusion avec celles de serveurs primaires et secondaires vues précédemment.

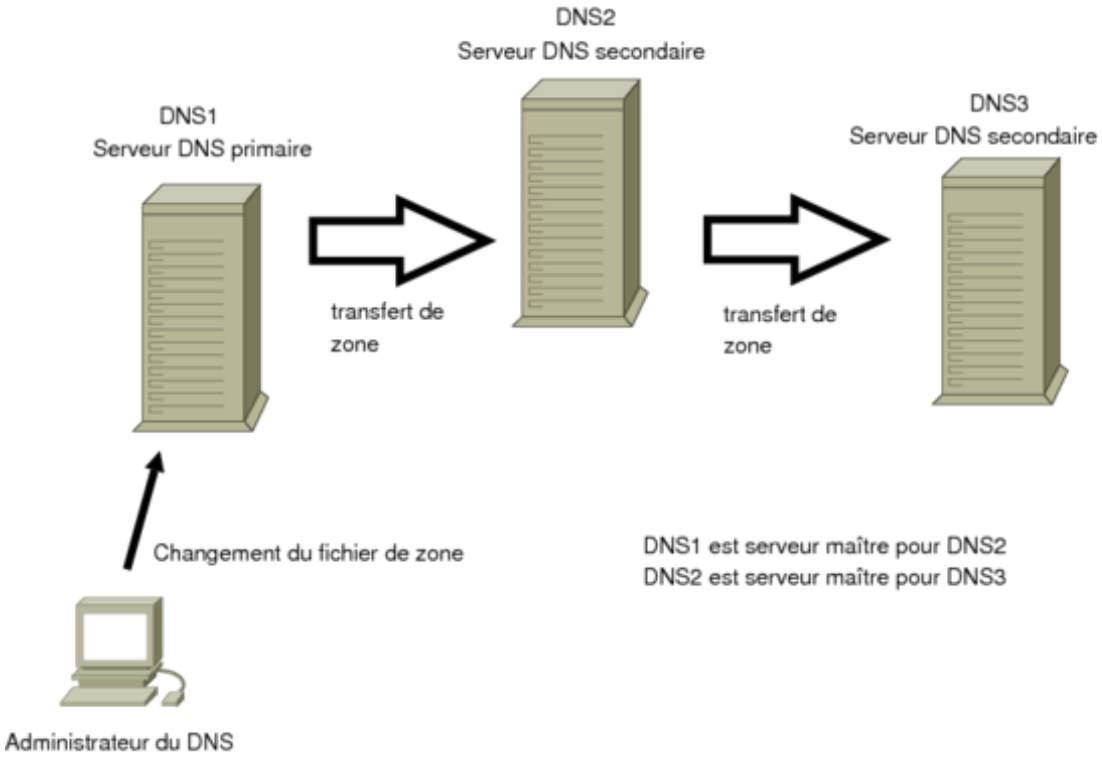



FIG. 2: Schéma simplifié des échanges de zones

Il s'agit bien d'une relation privée entre serveurs configurés pour fournir de l'information sur un même domaine (zone) et qui n'est pas communiquée au gestionnaire (*registrar*).

Cette notion de « maître/esclave » a été introduite dans la branche 8 de BIND et se rattache à une zone. Le principe est le suivant : une zone sur laquelle le serveur est maître correspond à des données de zone conservées localement, tandis qu'une zone esclave consiste à récupérer les données de zone sur un serveur DNS externe (« serveur maître »).

La définition accordée par Microsoft est différente, puisqu'un maître est toute source d'information pour un serveur secondaire, qu'elle soit locale ou distante (cf. schéma 2). Comme un serveur secondaire peut récupérer des informations pour sa zone depuis un autre serveur qui peut être primaire ou secondaire, un serveur secondaire peut donc être maître ou esclave. Plus d'informations techniques sont disponibles à l'adresse :

http://technet.microsoft.com/en-us/library/bb727007(TechNet.10).aspx

Un serveur, de la même manière qu'il peut être primaire ou secondaire pour une zone, peut être maître pour une zone et esclave d'autres différentes. Les serveurs maîtres et esclaves répondront comme ayant autorité sur les zones concernées.

Compte-tenu de la confusion dans l'utilisation de ces termes, nous ne les distinguerons pas dans la suite du document.

Il n'y a pas de restrictions directes sur le nombre de serveurs pouvant être maîtres pour une zone. Un exemple sous BIND :

```
zone "ex1.example.tld" in{
type master;
file "pub.ex1.example.tld";
};
zone "ex2.example.tld in{
type slave;
file "priv.ex2.example.tld";
masters{X.X.X.X;};
};
// file permet ici de stocker sur le disque les informations
// récupérées via le serveur maître d'adresse IP X.X.X.X
```

La question intuitive est donc alors de se demander quel moyen de communication est mis en œuvre pour communiquer entre maître et esclave afin d'échanger les données de zones. Il existe deux méthodes correspondant aux deux directions d'échanges :

- le serveur maître peut notifier (`NOTIFY`) les changements de zone. Cela garantit que les changements sont pris en compte très rapidement par les serveurs esclaves de la zone. Les opérations de type `NOTIFY` peuvent également être supprimées. Sous BIND, il suffit d'ajouter dans la déclaration de zone la variable `notify no`.
- le serveur esclave cherche spontanément à rafraîchir ses données quand le délai d'expiration pour l'enrgistrement SOA (paramètre *expiry*) est atteint. Si le maître est injoignable au moment du rafraichissement, le serveur esclave cessera de répondre.

Un serveur peut être « caché ». Pour cela, il s'agit de ne pas le déclarer auprès du *registrar* et de ne pas le référencer par des enregistrements NS dans les zones externes.

- un premier serveur est directement sur l'Internet (FAI par exemple) pour permettre l'accès à des services publics comme les sites Web, les serveurs mails ou ftp (DMZ) ;
- un serveur primaire caché se place dans la DMZ ou le réseau interne avec des règles de filtrage adaptées.

Dans cette architecture, le serveur public ne retourne que des réponses d'autorité et ne fait donc pas de cache ni n'autorise de requêtes récursives. Son fichier de zone n'inclut alors que les informations publiques (NS, MX, SOA, A, etc.).

Il faut alors prévoir de mettre à jour le serveur caché, bien qu'il ne soit pas déclaré dans les serveurs NS de la zone. Cela se fait sous BIND par le paramètre `also-notify`.

C'est un principe de redondance intéressant pour plusieurs raisons. En cas de panne matériel, il y a un équipement physique fonctionnel qui est rapidement disponible. En cas de compromission du système hébergeant le serveur DNS public, il n'y a pas de fuite d'information concernant la structure interne du réseau. Cette solution est donc plus complète qu'un seul partage de vue sur un même serveur.

## 2.9 Serveurs de redirection ou *forwarders*

La redirection peut servir à lier les résolutions de noms à un serveur particulier et ainsi à mieux contrôler le trafic DNS. Il s'agit d'un serveur configuré pour transférer toutes les requêtes vers un autre serveur et à les mettre en cache.

Il n'y a pas de modification particulière à faire pour transformer un serveur en service de redirection. En revanche, les autres serveurs doivent le connaître (cf. figure 3). Cela se traduit sous BIND par l'utilisation des paramètres `forward` et `forwarders` précisés pour une zone dédiée ou globalement.

Le mode de fonctionnement d'un serveur secondaire devient alors, suite à la réception d'une requête, le suivant :

- recherche dans les informations sur lesquelles il fait autorité ;
- si la recherche est négative, recherche dans son cache ;
- sinon, envoi de la requête vers la machine offrant le service de redirection.

Même si le serveur est configuré pour émettre une requête itérative, il enverra une requête récursive au serveur de redirection en espérant obtenir une réponse complète. C'est le mode *forward first*.

## 2.10 Apport de l'anycast

L'anycast (ou « envoi à la cantonade » selon le JO « Vocabulaire des télécommunications » du 28/12/2006) consiste à disposer de plusieurs serveurs rendant un service identique et laissant le client choisir le « meilleur ». Dans le cas du DNS, cette notion de « meilleur » correspond à la proximité IP et s'appuie sur les propriétés du protocole de routage BGP. Tous les sites disposant d'un même serveur DNS annoncent le même préfixe et chaque serveur dispose de la même adresse IP. L'algorithme de calculs de routes par BGP choisira ainsi l'instance qui servira la requête.

Il faut garder en tête une propriété importante évoquée dans l'un des premiers articles sur l'*anycast*, le RFC 1546 de novembre 1993 :

```
It is important to remember that anycasting is a stateless service.
An internetwork has no obligation to deliver two successive packets
sent to the same anycast address to the same host.
```

Cette méthode est également utilisée pour la diffusion distribuée de contenus (CDN, ou *Content Delivery Network*). Elle permet une plus grande résilience aux attaques de type déni de service.

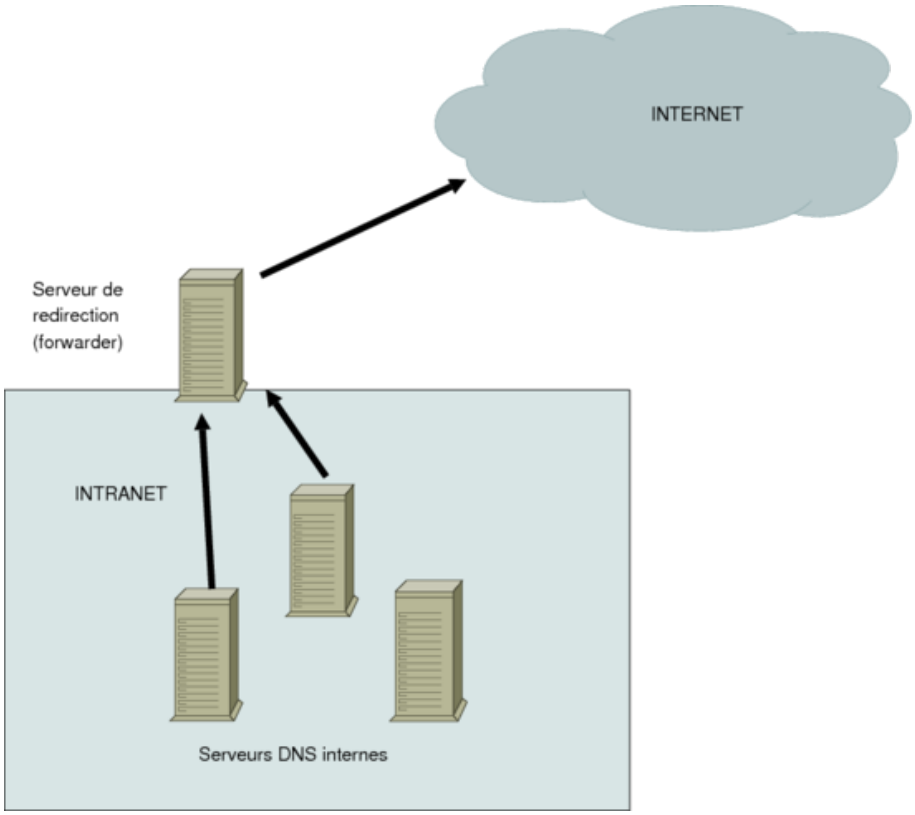



FIG. 3: Schéma simplifié de l'utilisation d'un serveur de redirection

Le DNS se prête bien à cette solution car il a la propriété d'être sans état. Les requêtes sont indépendantes en principe les unes des autres. Il faut en revanche s'assurer que les requêtes s'équilibrent bien entre les instances. Si l'une d'elles supporte tout le trafic, la solution perd de son intérêt.

La difficulté réside dans la surveillance des instances et dans leurs administrations, car, par construction, les instances ne doivent pas pouvoir être distinguées par les clients. Ainsi un dysfonctionnement de serveur n'est pas si simplement identifiable et corrigeable. Imaginons un serveur retournant une mauvaise réponse. Remonter à la machine à l'origine de l'erreur n'est pas assuré.

Des projets ont vu le jour pour surveiller les activités de chaque instance des serveurs racines, mais la qualité de leurs résultats dépend fortement des origines des requêtes et en particulier de la répartition des sondes émettrices.

- dnsprobe for probing anycast DNS analysis - Future Work (Wide Project)
  http://dnsmon.ripe.net
- CAIDA, "Measurements and Laboratory Simulations of the Upper DNS Hierarchy", 2004 :
  http://www.caida.org/outreach/papers/2004/dnspam
- TEAM CYMRU, "DNS Name Server Status Summary" :
  http://www.cymru.com/monitoring/dnssumm/
- CAIDA, "Nevil's root/gTLD DNS performance plots" :
  http://www.caida.org/cgi-bin/dns_perf/main.pl
- "NeTraMet in Wide Project" :
  http://dnstap.nc.u-tokyo.ac.jp/NeTraMet/
- D. Meyer, "University of Oregon Route Views Project" :
  http://www.routeviews.org/

Chaque instance répond au `hostbind.name`, seule distinction évidente.

```
Exemple de requête permettant de déterminer à première vue
l'instance du F-Root qui répond :

$dig +short +norec @192.5.5.241 hostname.bind chaos txt
```

```
"sfo2a.f.root-servers.org"
```

– Article de l'ISC, "A Sotware Approach to Distributing Requests for DNS Service using GNU Zebra, ISC Bind 9 and FreeBSD", mars 2004 :
http://www.isc.org/pubs/tn/isc-tn-2004-1.html

Fin 2007, le bilan est le suivant : il y a des réplications de serveurs racines dans 53 pays et 130 environnements physiques différents (il y a en réalité 13 serveurs « logiques » et plusieurs réplications pour ces derniers qui peuvent s'appuyer par exemple sur l'anycast présenté en section 2.10). La carte exacte n'est pas publique. Une carte représentant leur position approximative est visible à l'adresse suivante :

– Site *stupid.domain.name*, "DNS root servers in the world", septembre 2007 :
http://stupid.domaine.name/node/407

Ces serveurs ont des mises en œuvre différentes ; ils reposent en particulier sur BIND8, BIND9 ou NSD. Le matériel est hétérogène ainsi que les systèmes d'exploitation. L'effort pour maintenir la diversité est considéré avec attention par les différents opérateurs mondiaux.

Comme il est signalé dans le RFC 4786, la solidité de la solution anycast repose également sur BGP. Une annonce BGP pirate (RFC 4272) sera difficilement détectée. Or les usurpations BGP (aussi appelées *IP hijacking*) sont possibles et des incidents associés à cela ont été médiatisés en début d'année 2008. Le scénario n'est donc pas improbable.

Un historique relativement complet de l'aventure anycast associée au DNS est disponible à l'adresse :
http://www.av8.net/IETF-watch/DNSRootAnycast/History.html

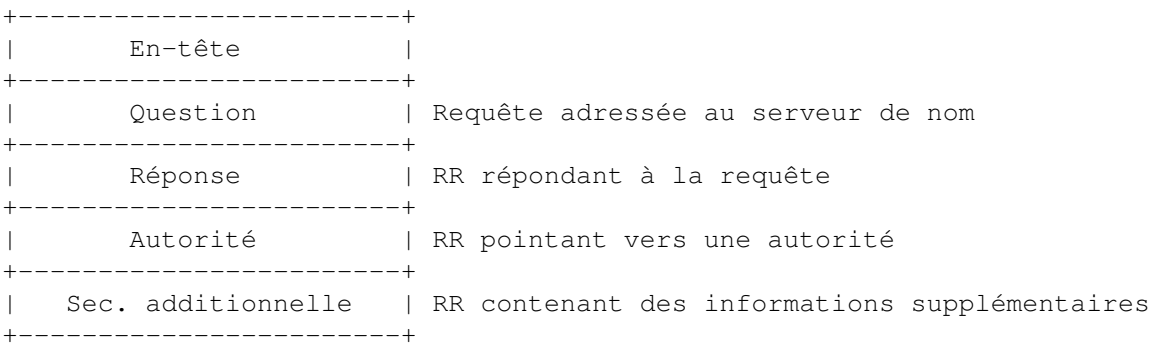
## 2.11 DNSv6 : quoi de neuf ?

Nous n'aborderons pas ici la version du DNS adaptée à l'adressage IPv6. Le lecteur trouvera les informations à ce sujet dans la note d'information CERTA-2006-INF-004, « Migrations IPv6, enjeux de sécurité ». La section 3.5 est consacrée à la continuité du service DNS dans sa variante DNSv6.

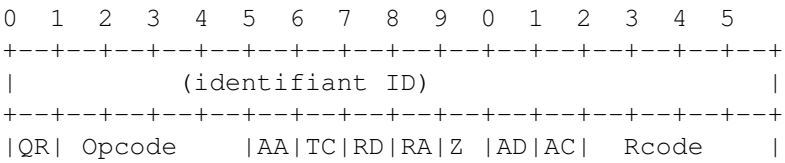
http://www.certa.ssi.gouv.fr/site/CERTA-2006-INF-004/

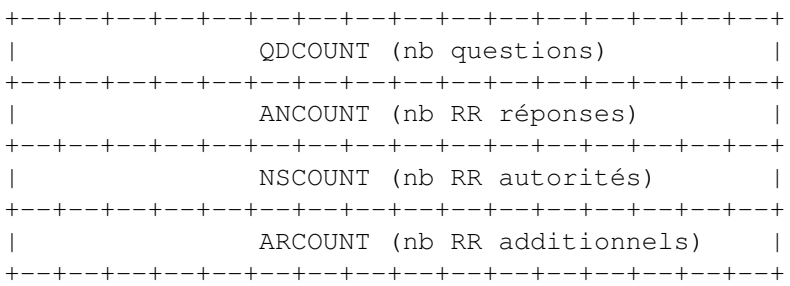
## 2.12 Format des messages DNS et trafic réseau

Les messages DNS transitent par les protocoles de transport TCP ou UDP. Le port associé est le 53. Ils ont le format général suivant :




L'en-tête d'une trame DNS a une longueur de 96 bits et a le format suivant :

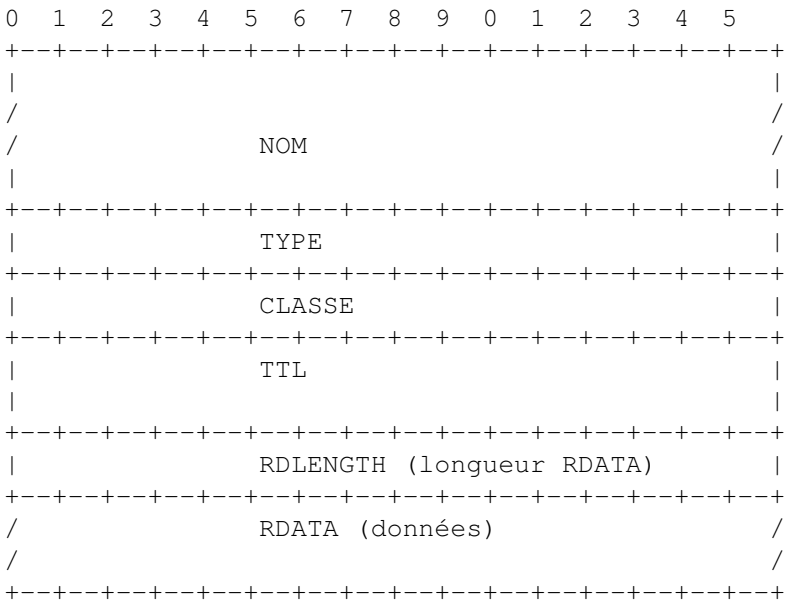





On retrouve les champs suivants (cf. RFC 1035) :

- un entier codé sur 16 bits représentant l'identifiant de la transaction DNS (requête/réponse) ou TXID ;
- un champ "QR" (1 bit) précisant s'il s'agit d'une requête ou d'une réponse ;
- un champ "Opcode" (4 bits) précisant le type de requête (standard, inverse, NOTIFY, état du serveur, etc.) ;
- un champ "Authoritative Answer" (1 bit) précisant si le serveur de noms qui répond a autorité pour le nom de domaine demandé ;
- un champ "TC" (1 bit) signalant si le message est coupé et donc que seuls les 512 premiers octets de la réponse sont inclus ;
- un champ "RD" (1 bit) signalé dans une requête et remis dans la réponse, pour préciser que le mode récursif est désiré ;
- un champ "RA" (1 bit) pour préciser si le serveur supporte le mode récursif ;
- un champ "Z" (3 bits) qui n'est pour l'instant pas utilisé de valeur nulle en principe. Le second bit peut indiquer si les données de réponses ont été authentifiées ;
- un champ précisant le code de retour, ou `Rcode` dans les réponses DNS. S'il vaut 0, aucune erreur n'est survenue ;
- quatre champs de 16 bits chacun fournissant le nombre d'entrées dans la section des questions DNS, le nombre d'enregistrements dans la section de réponse, le nombre de NS dans la section d'autorité et le nombre de RR éventuels fournis en complément dans la section additionnelle.

La suite du paquet contient ainsi zéro ou plusieurs sections (RR) dont le nombre d'entrées correspond pour chacune à la valeur des derniers champs de l'en-tête. Les enregistrements RR ont le format général suivant :




Chaque enregistrement, ou RR, est de taille variable et se compose d'un nom, d'un type, d'une classe ("IN" pour Internet bien souvent pour Internet), d'un TTL, d'un indicateur de la longueur des données Rdata et des données effectives.

Une liste des enregistrements possibles est maintenue à jour par l'IANA. Elle est disponible à l'adresse suivante :

– IANA, "Domain Name System (DNS) Parameters", avril 2008 :
  http://www.iana.org/assignments/dns-parameters

Les lignes suivantes montrent un exemple de requête et de réponse obtenus avec l'utilitaire `tshark` pour la résolution de `www.certa.ssi.gouv.fr` :

```
### REQUETE ####

Frame 1 (81 bytes on wire, 81 bytes captured)
(...)
User Datagram Protocol, Src Port: 32846 (32846), Dst Port: domain (53)
    Source port: 32846 (32846)
    Destination port: domain (53)
    Length: 47
    Checksum: 0x6f0e [correct]
        [Good Checksum: True]
        [Bad Checksum: False]
Domain Name System (query)
    Transaction ID: 0x0178
    Flags: 0x0100 (Standard query)
        0... .... .... .... = Response: Message is a query
        .000 0... .... .... = Opcode: Standard query (0)
        .... ..0. .... .... = Truncated: Message is not truncated
        .... ...1 .... .... = Recursion desired: Do query recursively
        .... .... .0.. .... = Z: reserved (0)
        .... .... ...0 .... = Non-authenticated data OK:
                              Non-authenticated data is unacceptable
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        www.certa.ssi.gouv.fr: type A, class IN
            Name: www.certa.ssi.gouv.fr
            Type: A (Host address)
            Class: IN (0x0001)


### REPONSE ###

Frame 2 (97 bytes on wire, 97 bytes captured)
(...)
User Datagram Protocol, Src Port: domain (53), Dst Port: 32846 (32846)
    Source port: domain (53)
    Destination port: 32846 (32846)
    Length: 63
    Checksum: 0x6c22 [correct]
        [Good Checksum: True]
        [Bad Checksum: False]
Domain Name System (response)
    [Request In: 1]
    [Time: 0.003499000 seconds]
    Transaction ID: 0x0178
    Flags: 0x8180 (Standard query response, No error)
        1... .... .... .... = Response: Message is a response
```

```
    .000 0... .... .... = Opcode: Standard query (0)
    .... .0.. .... .... = Authoritative: Server is not an authority for domain
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... 1... .... = Recursion available: Server can do recursive queries
    .... .... .0.. .... = Z: reserved (0)
    .... .... ..0. .... = Answer authenticated: Answer/authority portion was
                           not authenticated by the server
    .... .... .... 0000 = Reply code: No error (0)
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
Queries
    www.certa.ssi.gouv.fr: type A, class IN
        Name: www.certa.ssi.gouv.fr
        Type: A (Host address)
        Class: IN (0x0001)
Answers
    www.certa.ssi.gouv.fr: type A, class IN, addr 213.56.176.2
        Name: www.certa.ssi.gouv.fr
        Type: A (Host address)
        Class: IN (0x0001)
        Time to live: 22 minutes, 12 seconds
        Data length: 4
        Addr: 213.56.176.2
```

On peut remarquer dans l'exemple ci-dessus que le serveur qui a fourni la réponse n'a pas autorité. Il a fourni cette réponse depuis un cache.

Une extension de DNS, nommée EDNS (`Extended DNS`) est apparue en 1999 dans le standard RFC 2671. L'objectif de cette extension est de ne plus tenir rigueur de la limitation de taille imposée par les trames sous UDP dont le contenu est normalement de 512 octets maximum. Pour se faire, l'extension utilise des pseudo-type d'enregistrements appelés OPT. Un exemple démonstratif a été présenté par S. Bortzmeyer à l'adresse :

http://www.bortzmeyer.org/2671.html

L'exemple consiste à comparer les réponses pour interroger le TLD de Hong-Kong, qui serait l'un des plus importants en terme de serveurs de noms, avec les lignes de commande suivantes :

```
$dig NS hk.
   (retourne 15 enregistrements NS et 6 enregistrements additionnels)

$dig +bufsize=4096 NS hk.
   (retourne 15 enregistrements NS et 12 enregistrements additionnels)
```

## 2.13 Processus de résolutions locales

Il faut bien comprendre que la résolution de noms et l'usage du DNS ne sont nécessaires que si la correspondance entre nom et adresse IP n'est pas connue. Ce service n'est en théorie pas indispensable, mais en pratique si. Pour s'appuyer au minimum sur le service de résolution de noms DNS, les correspondances peuvent être notées « en dur » sur un poste afin d'éviter d'utiliser un tel trafic. Cette solution, qui est la technique historique de résolution de noms, est toujours fonctionnelle. Elle s'applique bien dans le cas où la liste des correspondants est bien définie et elle permet de s'abstenir de la résolution DNS qui n'est autre qu'un point de fragilité supplémentaire pour la bonne communication entre des machines.

La plupart des systèmes d'exploitation ont donc des fichiers locaux qui permettent de noter les correspondances entre adresses et noms de machines. Le processus de résolution utilisé se résume aux opérations suivantes :

1. recherche dans les informations stockées localement, par exemple dans le fichier :
   - /etc/hosts sous Linux. La séquence de recherche pour la résolution de noms est quant à elle précisée dans le fichier /etc/host.conf. L'entrée `multi on` par défaut précise que si plusieurs adresses sont présentes pour un même nom dans le fichier "hosts", alors tous seront retournés. Une autre option

intéressante est `nospoof on` qui vérifiera chaque résolution par une résolution inverse. Elle doit journaliser toute anomalie et donc fonctionne de pair avec l'entrée du fichier `alert` ;

- %WinDir%/system32/drivers/etc/hosts sous Windows ;

2. recherche dans un cache éventuel du système d'exploitation (cf. section 2.6 ;
3. recherche par requête DNS.

Ces fichiers locaux peuvent également être manipulés par des codes malveillants, comme cela est décrit dans la section 3.4.

# 3 Risques

Quelques risques ont été officiellement présentés en 2004 dans un document RFC 3833, ("Threat Analysis of the Domain Name System DNS") :
http://tools.ietf.org/html/rfc3833
On peut grossièrement répertorier cinq grandes familles :

- les atteintes à l'intégrité des données :
  - usurpation des réponses sur le réseau ;
  - pollution de cache visant à introduire des données incorrectes ;
  - modification malveillante des configurations (en particulier les postes clients/serveurs utilisant DHCP) ;
- les atteintes à la disponibilité : saturation du serveur (cache ou autorité) ou d'un équipement en coupure (routeur, pare-feu,...) par des trames ;
- l'utilisation des transferts de zone DNS pour collecter des informations sur l'architecture d'un système d'information et afin d'identifier des machines intéressantes (par exemple une machine nommée `serveur-test` ne sera peut-être pas très bien administrée) ;
- le contournement de la politique de sécurité en utilisant le système de requêtes/réponses DNS pour créer un tunnel caché ;
- des dépendances et des risques indirects (DNS racines, politique de mises en cache par les fournisseurs d'accès, etc.).

Chacune des sections suivantes développe plusieurs points de ces grandes familles.

## 3.1 Fabrication de fausses réponses

Cela consiste, pour un agresseur, à envoyer une réponse en lieu et place du serveur interrogé par un client. La très grande majorité des requêtes utilise le protocole de transport UDP, qui n'est pas orienté connexion. Le client est susceptible d'accepter une réponse illégitime si l'intrus connait ou anticipe quatre éléments :

1° l'instant où les échanges de trames seront pertinents ;
2° l'adresse du serveur de destination ;
3° le port source (qui sera donc le port destination de la réponse) ;
4° l'identifiant de la requête TXID (valeur aléatoire prévue par le protocole qui permet de distinguer plusieurs requêtes simultanées).

Enfin, il faut que la réponse leurre parvienne avant celle du serveur légitime, ce qui peut être obtenu en saturant ce dernier. Dans le cas où le protocole de transport est TCP (transferts de zone, par exemple), le numéro de séquence initial TCP (« ISN ») doit également être connu.

Lorsque l'agresseur est sur le même réseau local que le client ou le serveur, ces données peuvent être simplement récupérées par écoute du réseau, après une éventuelle attaque sur les équipements pour détourner/intercepter le trafic (attaques ARP, VLAN...).

Dans le cas contraire, le paramètre temps peut être relativement maîtrisé en incitant la victime à lancer une action (présentation d'un lien sur une page web, par exemple) qui nécessite une résolution de noms ; les valeurs du port source et de l'identifiant (TXID voire du numéro de séquence initial en TCP) doivent alors être estimées par divers moyens :

- lorsqu'il s'agit d'un cache, l'agresseur lui adresse préalablement des requêtes pour des ressources qu'il maîtrise et tente alors d'identifier l'évolution de ces valeurs ;
- envoyer une multitude de réponses pour augmenter la probabilité de tomber juste ;
- combiner les 2 moyens précédents...

Une génération aléatoire de qualité cryptographique du port source (un port source fixe est très mauvais) et de l'identifiant constitue une bonne méthode pour prévenir les agressions hors réseaux locaux.

Quant au transport TCP, le problème des systèmes d'exploitation avec des ISN prédictibles ne devrait plus être d'actualité.

## 3.2 Pollution des caches DNS

Puisque les clients légers se réfèrent à un cache (cf. section 2.6) et que ce dernier conserve les enregistrements pendant la durée qui lui a été spécifiée dans la réponse, un angle d'attaque consiste à introduire dans le cache des données sous contrôle de l'agresseur.

L'usurpation sur le réseau des réponses (voir section 3.1) faites au cache est un moyen d'action possible. Une autre possibilité réside dans la mauvaise gestion par certains logiciels caches (peut dépendre de la version) des réponses additionnelles (« glue records »). Les réponses additionnelles sont des informations qui sont transmises en complément de la réponse à la question posée.

L'objectif est encore la réduction du trafic DNS : ainsi, étant donnée la question « Quels sont les serveurs autorités pour le domaines `ssi.gouv.fr` ? », la réponse contiendra les noms des serveurs de la zone `ssi.gouv.fr` avec comme réponses additionnelles les adresses IP de ces serveurs, ce qui permet au client de les joindre directement sans émettre de nouvelles requêtes pour obtenir leurs adresses IP.

Si un agresseur amène un cache à interroger (en l'utilisant directement, en présentant un lien à un client légitime...) un serveur autorité sous son contrôle, il peut alors indûment désigner un serveur, qu'il souhaite usurper, comme autorité pour son domaine et donner une adresse IP arbitraire pour ce serveur dans les réponses additionnelles. Un cache mal conçu exploitera et gardera cette information additionnelle, bien qu'elle ne vienne pas d'un serveur ayant autorité sur le sujet et induira donc tous ses clients en erreur.

## 3.3 Généralités sur les redirections DNS

Les sections suivantes vont présenter quelques méthodes permettant à une personne malveillante de détourner les processus de résolution afin d'amener le poste de l'utilisateur à interpréter une réponse illégitime.

Quelles en sont les motivations ? Quelles sont alors les perspectives offertes aux personnes malveillantes ?

En premier lieu, il s'agit tout simplement de rediriger le trafic de l'utilisateur, partiellement ou totalement, vers des adresses contrôlées par la personne malveillante. Les raisons peuvent être simplement de l'interception de données, en redirigeant l'utilisateur vers l'adresse légitime ensuite. Un exemple consiste à rerouter les courriers électroniques vers un serveur ne correspondant pas à un enregistrement MX (serveur de messagerie) légitime. Il peut s'agir également de le faire interagir avec un faux service, afin de récupérer des données confidentielles (identifiants, mots de passe, etc.). Cette méthode peut également être utilisée pour du filoutage (*phishing*).

## 3.4 Les postes clients

Une méthode consiste à changer la configuration DNS du poste de l'utilisateur. Cela peut avoir lieu suite à une compromission directe de la machine ou peut résulter de la compromission d'un serveur fournissant ces informations. Cette seconde approche est détaillée dans la section suivante.

Plusieurs codes malveillants récents qui s'exécutent sur le poste de la machine changent des éléments locaux (cf. section 2.13) afin de modifier le processus de résolution de noms.

Les méthodes envisageables sont :

- ajouts d'entrées dans le fichier de données locales (*hosts*, cf. section 2.13) ;
- changement d'adresses IP des serveurs DNS dans la configuration réseau ;
- modifications des bibliothèques d'accès réseau.

Le CERTA a mentionné certains de ces codes dans des bulletins d'actualité :

- Bulletin d'actualité CERTA-2007-ACT-044, « Les modifications de configuration DNS » : http://www.certa.ssi.gouv.fr/site/CERTA-2007-ACT-044.pdf
- Bulletin d'actualité CERTA-2007-ACT-052, « Modification des fichiers » : http://www.certa.ssi.gouv.fr/site/CERTA-2007-ACT-052.pdf
- Bulletin d'actualité CERTA-2008-ACT-008, « Surveiller le trafic DNS, une nécessité » : http://www.certa.ssi.gouv.fr/site/CERTA-2008-ACT-008.pdf

## 3.5 Les serveurs DHCP et autres « box »

La configuration DNS est parfois fournie au moment de la négociation DHCP (option de code "6" définie dans le RFC 2132). Il faut donc à la personne malveillante trouver un moyen pour que le serveur DHCP fournisse une adresse IP non légitime.

Les boîtiers mis à disposition par les fournisseurs d'accès Internet ou les routeurs sont généralement configurables, en particulier par des interfaces Web. Les abonnés peuvent préciser les plages d'adresses à utiliser pour les baux ainsi que des serveurs DNS éventuels. Ces interfaces peuvent être rendues accessibles depuis l'Internet par le biais de code dynamique (Flash, JavaScript, etc.). L'utilisateur navigue sur une page contenant un code malveillant interprété par son navigateur et qui émet une requête de configuration à destination de l'interface Web interne du boîtier.

Le schéma 4 reprend les étapes principales de l'attaque :

1° l'utilisateur est amené à visiter une page Web (via un lien dans un courrier électronique, une redirection ou un lien contenu par une autre page par exemple) ;

2° la page Web contient un code malveillant qui est retourné au navigateur de l'utilisateur. Celui-ci l'interprète ;

3° le code cherche à se connecter à la page Web de configuration du routeur par son interface interne afin de modifier la configuration dont les informations DNS.

Ce genre d'attaque est facilitée par :

– des configurations laissées par défaut : l'adresse de l'interface Web interne, ainsi que les identifiants ou mots de passe sont relativement prévisibles ;
– l'existence de listes publiques des configurations par défaut ;
– une méconnaissance de la possibilité d'accéder à une interface interne via du code sur un site public et l'absence de règles de filtrage adaptées.

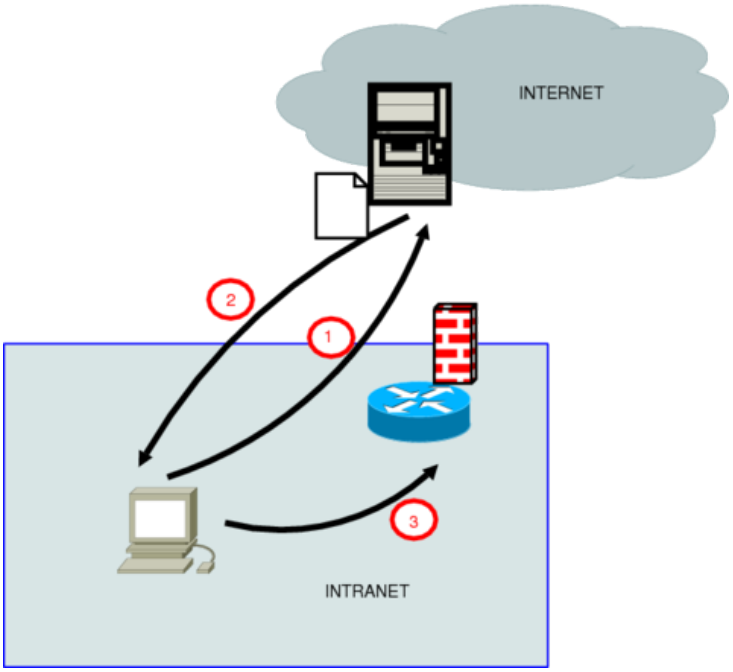



FIG. 4: Principe de modification malveillante de la configuration d'une « box »

Le lecteur est invité à consulter le bulletin d'actualité CERTA-2007-ACT-028. Ce dernier présente un article décrivant une méthode similaire nommée `dns pinning`.

http://www.certa.ssi.gouv.fr/site/CERTA-2007-ACT-028.pdf

## 3.6 Trames forgées ICMP

Comme il a été vu précédemment dans la section 2.12, les trames DNS n'ont pas une structure complexe, ce qui rend leur construction assez simple par des outils malveillants. Certaines vicissitudes permettent également

de simplifier les choses. À titre d'exemple, une propriété a été décrite dans le bulletin d'actualité CERTA-2008-ACT-026 : sous certaines conditions (prédictions des ports et des adresses), il est possible de forger des trames d'erreur ICMP (Type 3, Code 2) afin de perturber les communications avec les serveurs racines. Si les paquets légitimes sont suffisamment prédictibles, il suffit de construire une trame ICMP avec 64 bits de données « prévus » de la requête initiale. Il faut donc ajouter dans la trame ICMP un en-tête IP valide (20 octets sans option) et 64 bits de l'en-tête UDP, soit les ports source/destination, la longueur et la vérification (*checksum*). Si cette trame est correcte (il est toujours possible d'en émettre plusieurs), alors elle peut être interprétée comme un message d'erreur et provoquer l'interruption des échanges.

Plusieurs outils permettent de manière assez simple de construire et de tester de telles trames.

Une autre méthode consiste à tester les implémentations des couches protocolaires en envoyant des trames construites pseudo-aléatoirement. Il y a plusieurs méthodes dites de « *fuzzing* » qui permettent d'identifier des oublis ou des erreurs de codage dans les piles. Les conséquences peuvent être un déni de service voire une exécution de code arbitraire sur le système vulnérable.

Comme tout programme, le code d'un client ou d'un serveur DNS contient des vulnérabilités résiduelles dont le nombre, si la programmation respecte les règles de l'art, est fonction de la taille du code.

## 3.7 Amplification de trafic et dénis de service distribués

Les dénis de service contre l'infrastructure DNS ne sont pas rares. 7 des 13 serveurs racines DNS (cf. section 2.4) auraient été inaccessibles pendant plus d'une heure en 2002. Les médias se sont également faits écho d'une autre tentative en février 2007 mais qui n'aurait que très légèrement perturbé le fonctionnement global de l'Internet. Des serveurs DNS d'un groupe de diffusion distribuée de contenus (CDN) auraient été ciblés en 2004, provoquant l'inaccessibilité de sites comme Google, Yahoo ou Microsoft pendant plus de deux heures. Voici quelques liens mentionnant ces événements :

- Article de TheRegister, "DDoS attack really really tested", UltraDNS, 2002 : http://www.theregister.co.uk/2002/12/14/ddos_attack_really_really_tested/
- P. Vixie, G. Sneeringer, M. Schleifer, "Events of 21-Oct-2002", novembre 2002 : http://d.root-servers.org/october21.txt
- J. Hu, "Zombie' PCs caused Web outage, Akamai says", 2004 : http://news.zdnet.com/2100-1009_22-5236403.html
- R. Vaughn, G. Evron, "DNS Amplification Attacks", mars 2006 : http://www.isotf.org/news/DNS-Amplification-Attacks.pdf
- ICANN, "Factsheet - Root server attack on 6 February 2007" : http://www.icann.org/announcements/factsheet-dns-attack-08mar07.pdf
- ICANN, "SSAC Advisory SAC008 - DNS Distributed Denial of Service (DDoS) Attacks", mars 2006 : http://www.icann.org/en/committees/security/dns-ddos-advisory-21mar06.pdf

Des configurations particulières permettent à des personnes malveillantes d'amplifier des attaques de type DDoS. Elles profitent de la configuration récursive de certains serveurs mis en ligne sur le Web.

Pour ce faire, une personne malveillante crée un très grand nombre de requêtes récursives (bit demandant la récursivité de valeur "1" dans l'en-tête DNS) à des serveurs DNS autorisant la récursivité. Cet envoi peut être fait depuis un réseau de machines compromises, ou botnet, afin d'augmenter la dispersion des sources. Chaque trame envoyée usurpe la même adresse IP source. Les serveurs DNS répondent donc à cette machine.

L'attaquant y voit le bénéfice suivant : à partir de plusieurs trames émises vers diverses destinations, cela provoque en retour bien plus de trames reçues par la victime dont l'adresse a été usurpée. Des effets de bord sont également possibles, car les serveurs récursifs retransmettent également leur requête. Cet envoi de requête peut donc affecter de manière indirecte d'autres serveurs dont les racines.

L'un des principaux facteurs favorisant cette attaque est la taille des trames de réponses qui est souvent plus importante que celle des requêtes associées. Il suffit par exemple de profiter de l'extension DNS EDNS0 (RFC 2671) pour autoriser des paquets dépassant 500 octets.

Comment déterminer à l'avance la taille de la ressource interrogée ? Une possibilité est que l'attaquant a préalablement compromis un serveur DNS et a modifié le fichier de zone pour inclure son enregistrement de ressource (RR) de taille importante. L'attaquant collecte ensuite une liste de serveurs de noms qui émettront des requêtes récursives et retourneront sans contrôle particulier l'enregistrement récupéré.

Les ordres de grandeur sont intéressants. Dans certains incidents sur des serveurs racines (2006), le facteur d'amplification a pu atteindre 70 : une requête DNS représentant 60 octets de données a généré des réponses dépassant 4000 octets : 122 octets de réponses type 'A', 4000 octets de réponses TXT et 222 octets de réponse

SOA, soit 4320 octets. Comme ces réponses dépassent généralement le maximum autorisé (MTU), la trame peut être fragmentée au niveau IP.

L'affaire est d'autant plus intéressante que les serveurs récursifs ouverts ont également une fonction de cache. Cela évite dans ce cas de trop solliciter le serveur d'autorité hébergeant le fichier de zone compromis.

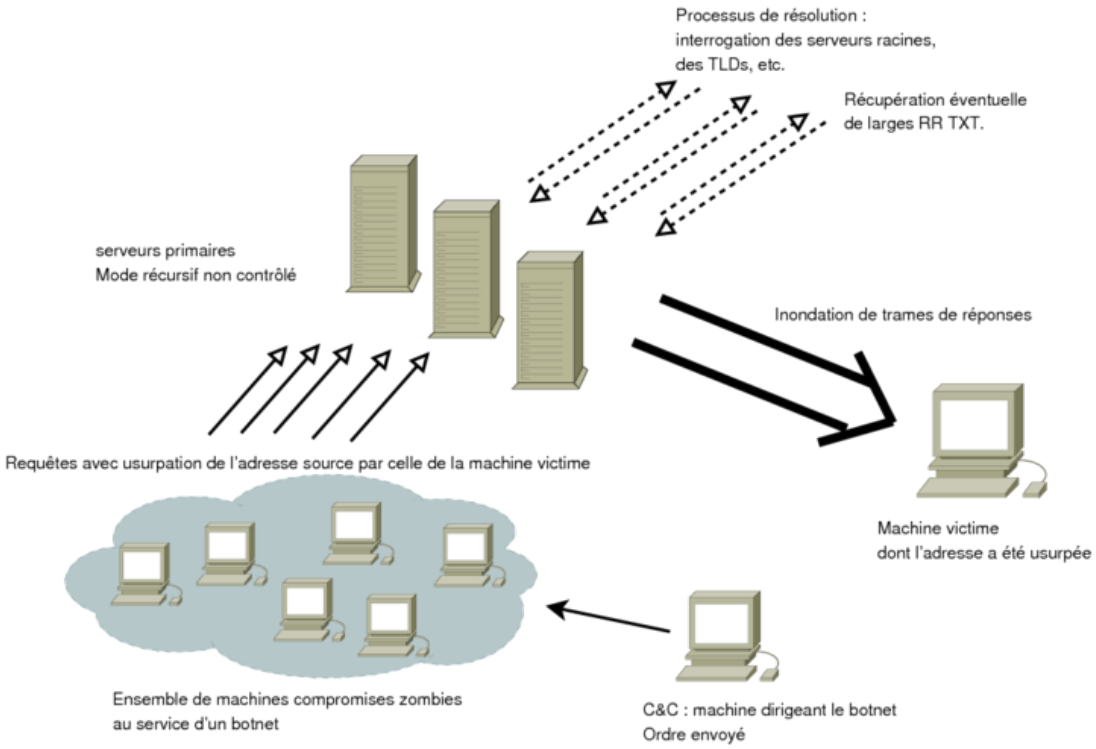



FIG. 5: Méthode d'amplification de trafic et déni de service

## 3.8 Informations fournies par les transferts de zone

Les transferts de zone (décrits dans la section 2.7 répétés nécessitent des ressources pouvant mettre à mal un serveur DNS. Outre ce problème, le transfert de zone en lui-même peut fournir une cartographie assez complète des systèmes utilisés dans un réseau. À titre d'exemple, cela peut permettre de récupérer des adresses et des noms comme pix001.monDomaine.tld, ou callmanager.monDomaine.tld. Une personne malveillante pourra donc assez simplement focaliser son attaque sur une adresse IP associée à un nom très évocateur du service fourni par la machine.

Comment vérifier si son serveur répond au transfert de zone ? Il peut suffire de tester la commande suivante :

```
$dig @SERVEUR_SERVEUR AXFR Domain.tld.
```

## 3.9 Risques indirects

### 3.9.1 Les serveurs racines

Des problèmes rencontrés début 2008 ont mis en cause certaines limites de la « gouvernance d'Internet ». Pour rappel, les serveurs racines présentés dans la section 2.4 sont responsables de la racine de la hiérarchie DNS. L'ICANN gère les adresses IP de ces serveurs. L'origine est le changement de l'adresse du serveur racine L (l.root-servers.net). L'adresse de ce serveur a été 198.32.64.12 jusqu'au 1er novembre 2007. Elle est alors devenue 199.7.83.42. De tels changements avaient eu lieu, par exemple en janvier 2004 avec le serveur racine B (b.root-servers.net). Le CERTA a rédigé un article dans son bulletin d'actualité :

– Bulletin d'actualité CERTA-2008-ACT-043, « Mise à jour des serveurs racines DNS » :
http://www.certa.ssi.gouv.fr/site/CERTA-2007-ACT-043.pdf

Ce changement d'adresse a provoqué l'annonce via BGP de l'ancienne adresse (198.32.64.0/24) par différentes entités : l'ICANN toujours (AS 20144) comme leur ancien serveur a été activé jusqu'au début du mois de mai 2008, mais aussi Community DNS (AS 42909) le 15 décembre 2007, ep.net (AS 4555) le 18 mars 2008 ou Diyixian.com (AS 9584) le 1er avril 2008.

Il s'agit d'un élément peut-être anecdotique, mais qui montre que les administrateurs réseau doivent rester vigilants sur les annonces et les mises à jour associées au DNS.

Une question qui vient naturellement à l'esprit alors est : comment les serveurs choisissent-ils le serveur racine parmi ceux qui sont retournés ? Est-ce le premier de la liste (A-Root) ou un choisi aléatoirement ?

Il n'y a pas de recommandation officielle dans les standards qui précise que les serveurs racines les plus prompts à répondre doivent être choisis.

Il y a globalement trois classes de méthodes différentes :

- les méthodes choisissant le meilleur serveur, la métrique choisie pouvant être le nombre de sauts, le délai aller-retour des trames, etc. ;
- les méthodes choisissant les serveurs pseudo-uniformément, soit par un système de rotation (round-robin) ou par décision aléatoire ;
- les méthodes utilisant différents serveurs, le choix étant pondéré selon des probabilités. Celles-ci se basent sur des métriques qui évoluent à chaque transaction afin d'offrir une meilleure adaptation aux changements des serveurs distants.

Dans la pratique, BIND (dans la version 9) utilise la dernière méthode, en s'appuyant sur la métrique du délai aller-retour d'un échange (ou RTT pour *Round-Trip Time*). A chaque fois qu'une requête est transmise à un serveur racine, le serveur note le temps mis pour recevoir une réponse. Ainsi, il choisit à chaque requête en fonction de celui qui a la plus courte valeur. Tant qu'aucune valeur RTT n'est connue, BIND lui attribue au hasard une valeur de RTT très courte afin de s'assurer qu'il ait les valeurs de RTT de l'ensemble des serveurs racines. En d'autres termes, il y a de fortes chances que le même serveur racine soit fréquemment utilisé si la métrique RTT présente peu de variations. L'algorithme utilisé est intéressant dans le cas où le serveur « élu » deviendrait plus difficilement joignable : il s'adapte et en choisit un nouveau. En revanche, il n'est pas capable de s'apercevoir automatiquement que les performances d'un autre serveur s'améliore.

Le serveur djbdns (TinyDNS) choisit quant à lui au hasard pour chaque requête à un serveur racine celui qui sera interrogé. Microsoft semble adopter la même pratique.

Les serveurs racines ne sont donc pas toujours choisis aléatoirement mais parfois par une méthode déterministe. L'utilisation quotidienne du DNS peut ainsi faire appel dans certaines conditions aux mêmes serveurs racines.

- M. L. Mueller, "Ruling the root. Internet governance and the taming of Cyberspace", mai 2002 : http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=8809
- R. Somegaza, K. Cho, Y. Sekiya, "The effects of server placement and server selection for Internet Services", 2002, IEICE : http://citeseer.ist.psu.edu/753444.html
- T. Lee, B. Huffaker, M. Fomenkov, K. Claffy, "On the problem of optimization of DNS servers'placement", 2003 : http://www.caida.org/outreach/papers/2003/dnsplacement/dnsplacement.pdf

### 3.9.2 Des chaînes de dépendance complexes dans la hiérarchie DNS

L'article référencé ci-dessous étudie les chaînes de dépendance qui sont créées par les délégations DNS. Un nom de domaine est dit « dépendant » d'un serveur de noms si celui-ci peut être impliqué dans la résolution du nom. Ces chaînes sont souvent très longues et peuvent inclure certains « maillons faibles » par lesquels il est envisageable de détourner des sous-domaines. L'analyse effectuée en 2005 montre ainsi que pour les informations collectées sur un peu moins de 600 000 domaines, le nombre de dépendances moyen est de l'ordre de 46 (cela varie finalement peu selon les TLD).

- E.G. Sirer, V. Ramsubramanian, "Perils of Transitive Trust in the Domain Name System", IMC 2005 : http://www.cs.cornell.edu/People/egs/papers/dnssurvey.pdf http://www.cs.cornell.edu/people/egs/beehive/

### 3.9.3 Résilience Internet et impacts indirects

Le CERTA a traité un incident intéressant relaté dans la section 1 de son bulletin d'actualité CERTA-2008-ACT-026 (« résilience DNS et déni de service »). Le serveur d'autorité (SOA) d'une organisation était injoignable.

En revanche, un autre serveur NS pouvait répondre pour les domaines concernés. L'analyse a montré que la résolution fonctionnait différemment selon les serveurs de différents FAI interrogés : tantôt elle retournait une réponse correcte, tantôt elle n'aboutissait pas. Certains serveurs DNS mettant donc en cache les enregistrements de zones pour lesquelles ils ne sont pas autorités ne le font qu'en se basant sur un seul des serveurs DNS ayant autorité de ces zones. Si ce dernier est indisponible, les caches ne consultent pas les autres NS pour obtenir des enregistrements. À l'expiration du délai de conservation en cache, les domaines sont considérés comme inexistants par ces serveurs de cache.

La constatation d'un déni de service est donc délicate : il peut arriver que ce soit un équipement de l'observateur lui-même, ou la chaîne de recherche de l'information qui soit à l'origine du déni de service. Seul le recoupement entre plusieurs points d'observation et de mesure permet d'identifier la source du problème.

## 3.10 Détournement du protocole et notion de canaux cachés

Il y a plusieurs façons de dissimuler des informations à tout niveau des différentes couches protocolaires. L'objet n'est pas ici de passer certaines de ses solutions connues en revue, mais de s'attarder sur le détournement de fonctionnalité qui peut être fait des trames DNS. Pour un panorama plus complet des tunnels, le lecteur peut se reporter à la note d'information CERTA-2001-INF-003, « Tunnels et pare-feux : une cohabitation difficile ».

Les motifs : certaines politiques de sécurité précisent qu'un utilisateur doit s'authentifier avant d'accéder à une quelconque ressource. C'est le cas par exemple de certains portails captifs en Wi-Fi qui redirigent le visiteur par défaut sur une page de connexion. Les règles de filtrage oublient parfois par facilité de considérer le trafic DNS. Or ce n'est pas toujours le service légitime associé à un port (IANA) qui est utilisé pour ce port. En d'autres termes, autoriser tout flux entrant et sortant sur le port 53/udp ne garantit aucunement que seul du trafic DNS légitime transite par ce port.

La méthode consiste alors à exploiter des champs du protocole DNS pour échanger des informations avec un serveur distant contrôlé correctement. Cette méthode a été abordée dans le bulletin d'actualité CERTA-2007-ACT-051.

Il faut au préalable contrôler un domaine, par exemple *monDomaine.tld* : le serveur DNS distant donne délégation pour un des sous-domaines, *tunnel.monDomaine.tld* a une machine appelée ici *tunnelHostSrv*. Cela est possible en ajoutant quelques lignes simples dans le fichier de zone du serveur en charge de *monDomaine.tld* :

```
tunnelHostSrv IN  A  X.X.X.X
tunnel IN NS  tunnelHostSrv.monDomaine.tld.
```

La machine *tunnelHostSrv* est en écoute sur son port 53/udp. Elle met en œuvre des petits scripts disponibles librement qui s'appuient très souvent sur les pilotes TUN/TAP (matériels *Virtual Point-to-Point* et *Ethernet*) afin de transformer les données reçues par la carte réseau et fournir une connectivité au niveau IP. Le client communique ensuite avec cette machine, soit directement, soit via le serveur local faisant relais pour les requêtes DNS.

Côté requête, l'ajout de données n'est pas si évident. La possibilité la plus courante est de chiffrer et/ou brouiller les données dans le nom à résoudre, en codant les données sous une forme textuelle comme Base32 (ou Base64).

Dans les réponses, les enregistrements TXT et NULL permettent d'envoyer de grosses quantités d'informations. Ils peuvent cependant être filtrés par les serveurs DNS. Un exemple de tunnel est présenté par la figure 6.

Enfin, certains tunnels profitent de l'extension EDNS0 pour augmenter la taille des réponses.

Des outils ont également profité des enregistrements CNAME plus fréquemment rencontrés que TXT ou NULL.

Les données échangées peuvent également être brouillées (compression gzip, codage en base32, etc.). Un tunnel SSH peut aussi être utilisé par dessus le tunnel DNS précédemment construit. Une option intéressante de SSH est dans ce cas la compression de données avec l'option `-C`.

Les paquets IP initiaux à envoyer sont donc découpés en plusieurs paquets DNS puis émis séparément. Ces derniers sont alors recomposés côté serveur (propriété de la fragmentation IP). Une autre façon est de réduire le MTU des interfaces du tunnel afin que le système d'exploitation prenne directement en considération les aspects de fragmentation. Le tunnel créé fournit un canal au-dessus d'un protocole n'offrant aucune garantie et pouvant donc engendrer des pertes.

- Bulletin d'actualité CERTA-2007-ACT-051 du 21 décembre 2007, section 6, « Les canaux cachés DNS » : http://www.certa.ssi.gouv.fr/site/CERTA-2007-ACT-051.pdf
- «Tunnels et pare-feux : une cohabitation difficile », Note d'information CERTA-2001-INF-003 du 29 août 2001 :
  http://www.certa.ssi.gouv.fr/site/CERTA-2001-INF-003/

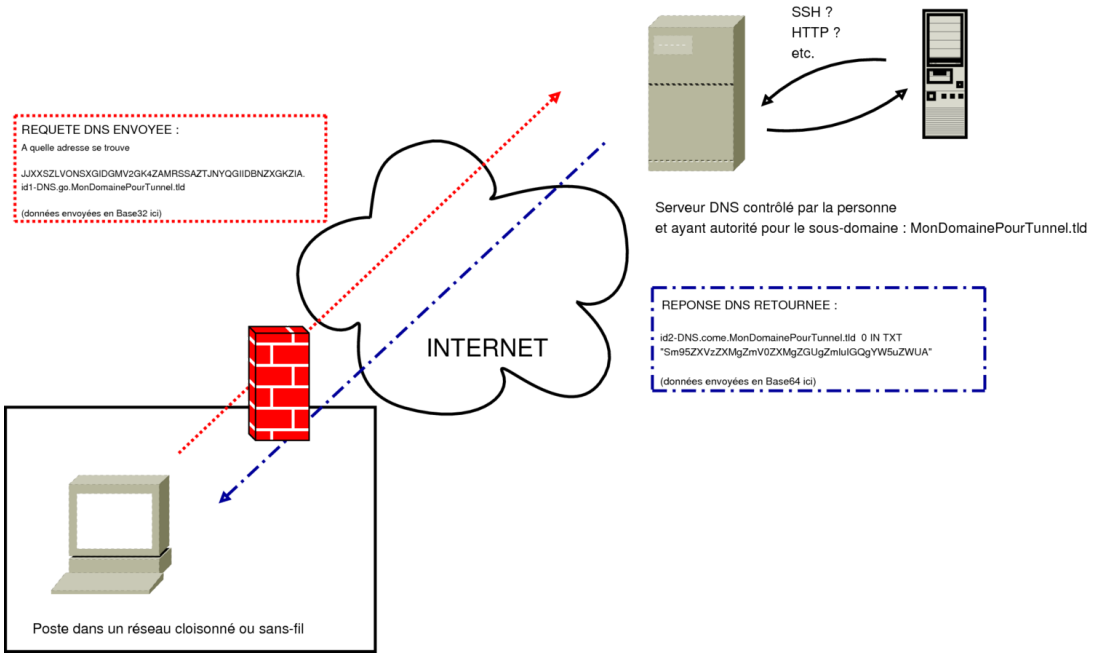



FIG. 6: Schéma d'un tunnel DNS possible

### 3.11 Récapitulatif des vulnérabilités signalées par le CERTA en 2007 et 2008

Le tableau 2 résume les vulnérabilités recensées par le CERTA en 2007 et 2008 et relatives au DNS.

| Référence CERTA | Titre de l'avis | Date de parution |
|---|---|---|
| CERTA-2007-AVI-056 | Vulnérabilité du serveur DNS BIND | 26/01/2007 |
| CERTA-2007-AVI-199 | Vulnérabilité de BIND | 02/05/2007 |
| CERTA-2007-AVI-209 | Vulnérabilité RPC de Microsoft DNS Server | 09/05/2007 |
| CERTA-2007-AVI-327 | Vulnérabilité dans BIND | 24/07/2007 |
| CERTA-2007-AVI-490 | Vulnérabilité du serveur DNS de Microsoft Windows | 14/11/2007 |
| CERTA-2008-AVI-039 | Vulnérabilité dans ISC BIND | 28/01/2008 |
| CERTA-2008-AVI-144 | Vulnérabilité de la bibliothèque Perl Net::DNS | 19/03/2008 |
| CERTA-2008-AVI-191 | Vulnérabilité du client DNS de Microsoft Windows | 09/04/2008 |
| CERTA-2008-AVI-353 | Vulnérabilité DNS dans Microsoft Windows | 09/07/2008 |
| CERTA-2008-AVI-358 | Vulnérabilité dans les produits Cisco | 09/07/2008 |
| CERTA-2008-AVI-359 | Vulnérabilité dans ISC BIND | 09/07/2008 |
| CERTA-2008-AVI-360 | Vulnérabilité dans l'implémentation DNS de Juniper | 09/07/2008 |

TAB. 2: Quelques avis du CERTA concernant DNS

On notera en particulier deux problèmes distincts corrigés par plusieurs éditeurs en 2007 et 2008 :

- le premier concerne la génération des identifiants de transactions qui n'est pas suffisamment aléatoires ; http://www.certa.ssi.gouv.fr/site/CERTA-2008-ACT-028.pdf
- le second concerne le choix des ports sources utilisés pour chaque transaction et qui ne seraient pas suffisamment aléatoire (voire constants). http://www.certa.ssi.gouv.fr/site/CERTA-2008-ACT-028.pdf

# 4 Configurations recommandées et préconisations

## 4.1 Introduction

Plusieurs risques pesant sur l'architecture DNS ont été présentés dans les précédentes sections. Celle-ci a pour objectif de présenter différentes contre-mesures. Elles sont applicables à différentes échelles, par des administrateurs de réseau, des responsables du service DNS ou des offices d'enregistrement.

Les principes généraux sont les suivants :

- mettre à jour les logiciels offrant les différents services ainsi que les systèmes les hébergeant ;
- séparer logiquement et physiquement les différents acteurs DNS du réseau : en l'occurrence les caches, les serveurs de résolution internes récursifs, les serveurs d'autorité de zone publics, les serveurs miroirs ;
- adapter les fichiers de configuration et tester leur cohérence ;
- déployer autant que possible des méthodes garantissant l'authentification et l'intégrité des échanges d'information (DNSSEC, TSIG, etc.) ;
- appliquer des règles de filtrage strictes pour les communications DNS ;
- surveiller le trafic.

Chacun de ces points est repris dans les paragraphes suivants.

## 4.2 Localisations physique et logique des serveurs

Les sections 2.3, 2.5 et 2.8 ont montré que les serveurs DNS pouvaient avoir différents rôles, et, par conséquent, différents modes de configuration. Ces différences sont complétées par des distinctions physiques et logiques.

Les serveurs de noms doivent être physiquement dispersés. Il faut prévoir au minimum un second site d'hébergement. Cette précaution reste valable au niveau de l'architecture du réseau. Il faut s'assurer que tous les serveurs de noms ne dépendent pas d'un même élément physique (routeur, switch, etc.), ne soient pas dans le même sous-réseau et ne soient pas sur une même ligne dédiée.

Plutôt que de mettre en service un seul ensemble de serveurs pour tous les types de « clients », il est préférable de disposer d'ensembles (architectures) distincts. Un premier ensemble externe peut être placé dans une DMZ. Il s'agira des serveurs de noms qui ne rendront que les enregistrements RR pertinents aux machines externes, c'est-à-dire celles avec des services publics (serveurs Web, serveurs de messagerie, etc.). L'autre ensemble, interne, doit être localisé derrière un pare-feu afin de garantir qu'il ne soit pas atteignable depuis l'extérieur et qu'il offre des services de résolution de noms aux seules machines internes.

Pour une architecture externe, il est préférable de placer le serveur primaire dans une zone du réseau protégée par un filtrage strict. Les règles de filtrage doivent permettre aux serveurs secondaires de l'interroger et d'effectuer des transferts de zone. Tout autre trafic est bloqué. Ce serveur primaire peut également être en mode caché. Il n'apparaîtra donc pas dans les enregistrements NS. Les serveurs secondaires sont eux placés dans la DMZ ou externalisés. Le serveur de redirection a lui pour objectif de gérer toutes les connexions DNS des utilisateurs internes. Il se trouve comme le serveur primaire, derrière un pare-feu et communique exclusivement par DNS avec les serveurs de noms Internet.

L'intérêt de distinguer ces deux ensembles est d'éviter d'exposer les services internes inutilement. Cela s'appelle le *split DNS*.

Dans la mesure du possible, ce cloisonnement physique est préférable aux propriétés de « vues » offertes par certains serveurs comme BIND.

Il n'y a pas la même confiance apportée à un serveur répondant à des requêtes externes et un relayant les demandes de machines internes. Il convient donc de séparer physiquement ces fonctions.

Certaines pratiques consistent à installer « deux serveurs en un », si les ressources matérielles sont limitées. La séparation est alors opérée par le logiciel utilisé par le biais d'un fichier de configuration unique.

Une méthode plus propre si le matériel se fait rare consiste à utiliser deux processus distincts. Mais il n'est pas possible d'indiquer à des serveurs ou des clients distants d'interroger le service sur un autre port que le 53, il faut exécuter ces serveurs sur des adresses IP distinctes (l'idéal étant de disposer de plusieurs interfaces réseau dans ce cas).

Le lecteur comprendra également après les remarques précédentes que des boitiers trouvés dans le commerce et multipliant les fonctionnalités ne sont pas un choix adapté. Certains constructeurs proposent ainsi des routeurs faisant aussi office de serveur DNS ou gérant de redirections DNS (*forwarding proxy DNS Server*). Les risques diminuent et la maintenance se trouve facilitée si les éléments ne gèrent qu'une seule fonctionnalité.

## 4.3 Filtrage

### 4.3.1 Principes

Il est nécessaire d'avoir une bonne compréhension du DNS et des caractéristiques du serveur afin de configurer correctement des mesures de filtrage. Celles-ci peuvent être faites au niveau des trames de réseau/transport ou directement au niveau applicatif. Ces cas de filtrage sont détaillés dans les paragraphes suivants.

### 4.3.2 Filtrage au niveau réseau

Les bonnes pratiques de filtrage définies dans la note d'information CERTA-2006-INF-001 s'appliquent. Le DNS utilise intensivement UDP.

Il est important d'effectuer un filtrage par adresse source pour limiter les risques d'usurpation. Une méthode consiste pour chaque fournisseur d'accès à vérifier les adresses IP en périphérie interne de leur réseau. La règle de base au niveau du routeur de périphérie recevant une trame sortante est :

```
SI
          l'adresse IP source reçue provient bien
           d'une plage d'adresses internes allouée
ALORS
          je transmets le paquet
SINON
          je le jette
```

L'inconvénient principal de cette méthode est qu'elle nécessite la bonne volonté d'une majorité d'acteurs. Elle est également mal adaptée pour certains environnements mobiles (ou *multihoming*) et n'empêche pas des usurpations au sein d'un même réseau.

- RFC 2827, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", mai 2000 :
  http://www.ietf.org/rfc/rfc2827.txt
- P. Vixie, ICANN, SSAC004, "Securing the Edge", octobre 2002 :
  http://www.icann.org/en/committees/security/sac004.txt

Les échanges DNS entre tous les acteurs de l'architecture DNS doivent être rigoureusement filtrés. Les listes de contrôles d'accès qui peuvent être appliqués au niveau du serveur doivent être appliqués au niveau du pare-feu également.

Par exemple, si la vue d'une zone se limite à une plage d'adresses internes, alors il faut prévoir dans le pare-feu de séparation une règle de la forme :

```
Politique générale : REJET

sauf

Sens|    IP         |  IP           | Proto|Port| Port |
    |    src        |  dest         |      |src | dest |
-------------------------------------------------------------

IN  |192.168.0.0/24| 192.168.1.3   | UDP  |>1023| 53  |   ACCEPT

OUT | 192.168.1.3   |192.168.0.0/24| UDP  | 53  |>1023|   ACCEPT
```

Cela restreint l'accès DNS du réseau 192.168.0.0/24 à l'unique machine 192.168.1.3. Réciproquement, celle-ci ne peut envoyer que certaines trames de réponses. Ce type de règle doit être défini et rédigé sur l'ensemble des éléments filtrants.

Cette procédure peut être fastidieuse pour des architectures DNS complexes mais elle reste indispensable.

D'autres règles de filtrage au niveau réseau ne doivent pas être négligées. Il faut par exemple s'assurer que les trames ICMP sont bien jetées. Le CERTA a rédigé à ce sujet un article dans son bulletin d'actualité CERTA-2008-ACT-026 (section 2 « Des protocoles anodins ? »). Sous certaines conditions (prédictions des ports et adresses), il est possible de forger des trames d'erreur (Type 3, Code 2) afin de perturber les communications avec les serveurs. Cela a été évoqué dans la section 3.6.

Une bonne pratique consiste à configurer un pare-feu local sur le serveur DNS. Les dernières mises à jour datant du mois de juillet 2008 de BIND consistent à ajouter de l'aléa dans chaque port source choisi pour construire les requêtes. Une autre façon de procéder est d'appliquer une translation de ports au niveau du pare-feu. Cela peut se faire sous *netfilter/iptables* ou *Packet Filter*. Voici un exemple (à adapter) :

```
NETFILTER/IPTABLES (UDP et TCP)
  =>
iptables -t nat -A POSTROUTING -o ethX -j SNAT -p udp -s Y.Y.Y.Y --dport 53
                                                    --to X.X.X.X --random

PACKET FILTER - PF
  =>
nat on dcX inet proto { tcp, udp } from Y.Y.Y.Y to any port 53 -> X.X.X.X
```

Sous Netfilter/iptables, la valeur aléatoire du port ne change pas à chaque paquet mais selon les éléments (IP src/IP dst /port src/port dst/ protocole) dans une fenêtre de temps déterminée (quelques dizaines de secondes). Sous PF, les ports source sont choisis au hasard de manière implicite, dès que la translation NAT est utilisée.

#### 4.3.3 Piles protocolaires des serveurs

Il faut s'assurer que la mise en œuvre des protocoles réseau des serveurs est adaptée aux risques d'attaques par déni de service. Certains systèmes permettent de raffiner la configuration de la pile en jouant sur :

1. la taille de la « fenêtre » TCP quantifiant la taille des échanges possibles (RFC 1323) ;
2. la longueur des files d'attentes pour les créations de connexions TCP. Une alternative est l'utilisation de TCP *SYN cookies* ;
3. les redirections IP qui doivent être bloquées ;
4. le temps d'attente maximal d'une session TCP (TIME_WAIT) ;
5. le délai de mise en cache ARP ;
6. les réponses aux échanges ICMP diffusés (*broadcast*), ainsi que les autres trames qui ne sont pas directement adressées à une machine ;
7. etc.

Il est possible sur certaines distributions comme Linux de modifier les variables système afin d'imposer un choix aléatoire des ports. Ce changement peut s'effectuer avec la commande `sysctl` :

```
$sysctl net.inet.ip.portrange.randomized
net.inet.ip.portrange.randomized: 1
$sysctl -d net.inet.ip.portrange.randomized
net.inet.ip.portrange.randomized: Enable random port allocation
```

#### 4.3.4 Unicast RPF

Le RFC 3704 décrit un autre concept pour limiter les usurpations de trames qui relève du bon sens mais qui n'est pas toujours mis en œuvre par défaut : le *Unicast Reverse Path Forwarding*. Les interfaces ne doivent pas accepter toute adresse émettrice indifféremment. L'idée initiale était bien de ne pas propager des adresses privées sur l'Internet, mais il peut être étendu au principe suivant : en s'appuyant sur les données de la table de routage, les paquets ne sont transférés que s'ils sont émis depuis la meilleure route estimée de la source trouvée. Cette solution ne fonctionne pas dans tous les cas (routes asymétriques, route par défaut). Voici quelques méthodes pour la mettre en œuvre :

- Sous Cisco

```
uRPF strict mode
interface GigabitEthernet0/0
 ip verify unicast source reachable-via rx
```

- Sous Juniper

```
interface { ge-0/0/0 { unit 0 { family inet {
 rpf-check;
 }}}}
```

- Sous PF (Packet Filtering)

```
block in quick from urpf-failed label uRPF
```

### 4.3.5 Contrôle des transactions DNS

La politique de filtrage classique s'applique au niveau applicatif. Il ne faut autoriser que les flux et les transactions qui sont légitimes. Cette procédure peut permettre d'éviter des compromissions de cache, de détecter certaines compromissions de postes ou de limiter des fuites d'informations.

Le trafic DNS sortant d'un réseau ne doit être autorisé que pour les serveurs internes. En d'autres termes, tout utilisateur qui changerait de serveur DNS pour en utiliser un extérieur (suite à une compromission ou intentionnellement par méconnaissance des risques) ne devrait pas avoir de requêtes qui aboutissent. Le filtrage des connexions sortantes doit être journalisé et attentivement analysé car c'est un excellent moyen de détecter toute violation de la politique de filtrage et toute compromission potentielle. Le pré-requis reste ici la mise en place d'une politique de filtrage des connexions sortantes.

Le port de destination pour un serveur utilisé pour l'envoi des datagrammes en UDP ou TCP est 53. Les datagrammes DNS en UDP sont limités à 512 octets (valeur représentant les données sans l'entête UDP et IP). Les datagrammes plus longs doivent être tronqués à l'aide du champ TC.

L'utilisation d'UDP n'est pas recommandée pour les transferts de zone, mais uniquement pour les requêtes standards. L'utilisation de TCP peut être revue dans certains cas étant plus robuste aux usurpations de trames.

Certains enregistrements sont bien définis par les standards mais peuvent être inutiles dans plusieurs environnements. Le seul impact de les inclure dans les fichiers de zone consiste à permettre à une personne malveillante d'y accéder et de récupérer des informations complémentaires. Parmi les enregistrements qui sont facultatifs, il y a :

- HINFO : cet enregistrement fournit des informations sur l'hôte, comme le type de processeur et le système d'exploitation (RFC 1700) ;
- RP : cet enregistrement identifie la personne ou le groupe responsable de la machine hôte ;
- LOC : cet enregistrement fournit pour la zone des détails géographiques (RFC 1876), comme la latitude et la longitude ;
- etc.

De nombreuses initiatives ont cherché à exploiter l'architecture DNS pour véhiculer d'autres services que la résolution de noms (listes de serveurs, distributions de clés publiques…). Celles-ci ont parfois abouti à la création de RR qui ne sont pas ou peu utilisés.

Comme tout filtrage, la meilleure pratique consiste à bloquer tout type de requête sauf celles les enregistrements prédéterminés. Il faut également vérifier les fichiers de zone pour s'assurer que ces RR n'apparaissent pas (si inutiles).

Un standard précise les attitudes que les serveurs doivent adopter quand un enregistrement n'est pas reconnu :

- RFC 3597, "Handling of Unknown DNS Resource Record (RR) Types", septembre 2003 : http://www.ietf.org/rfc/rfc3597.txt

Il y est écrit que le serveur ne doit pas modifier les données RDATA méconnues et doit rediriger celles-ci telles quelles. En pratique, ce principe est difficilement applicable et présente plus de risques que d'avantages.

Au niveau des transferts de zone (voir section 2.7, un serveur secondaire ne lance pas directement le processus à la réception d'une trame de notification (NOTIFY). Il envoie plusieurs trames dans l'ordre suivant :

- réponse au message NOTIFY envoyé par le serveur primaire ;
- demande de l'enregistrement SOA de la zone dont le numéro de série a changé ;
- lancement du transfert de zone (si le numéro de série récupéré annonce bien une modification).

Comme il vient d'être dit, le serveur secondaire ne lance donc pas directement un transfert de zone à la réception du message NOTIFY. La vérification de ce processus permet d'identifier certaines fausses annonces NOTIFY qui auraient pu être forgées.

Il est important de surveiller les échanges NOTIFY entre les serveurs afin de détecter toute tentative d'abus.

De manière générale, toute trame qui ne correspond pas à la configuration doit être rejetée. En particulier, si le serveur de noms n'est pas récursif, il faut prévoir un filtrage sur les bits "RA" (*Recursion Available*).

Il faut également filtrer les trames malformées ou contenant des adresses erronées (*bogons*).

http://www.cymru.com/Documents/bogon-list.html

Les options EDNS0 doivent être filtrées si elles ne sont pas nécessaires.

Comme le CERTA l'a mentionné dans sa note d'information CERTA-2006-INF-002, IPv6 est maintenant omniprésent, que l'on veuille l'adopter ou non. Le volume de requêtes AAAA ne cesse d'augmenter et l'utilisation involontaire d'IPv6 a pour conséquence de doubler le volume du nombre de requêtes DNS : une première requête de type AAAA n'aboutissant pas, une seconde requête de type A est émise. Il faut donc vérifier que les piles protocolaires sont bien désactivées quand elles ne sont pas nécessaires. Le lecteur trouvera un ensemble d'indications à l'adresse :

http://www.certa.ssi.gouv.fr/site/CERTA-2006-INF-002/

Enfin, il peut être intéressant d'appliquer des limitations volumétriques par adresse source, que ce soit en terme de débit ou en terme de trames par unité de temps.

## 4.4 Authentification, Confidentialité et Intégrité des échanges : solutions ?

### 4.4.1 « Signatures » des transactions : TSIG

Cette méthode est initialement décrite dans le RFC 2845. Il s'agit d'une « signature » à clé secrète (algorithme de hashé actuel HMAC-MD5, non inversible), utilisée pour effectuer des transferts de zone, des mises à jour dynamiques, ou des transactions entre le *resolver* et la machine de cache. Ces opérations sont effectuées entre serveurs connus et très souvent maîtrisés (dans la même administration ou organisation), ce qui a motivé le choix d'utiliser un secret partagé. Celui-ci est donc utilisé pour l'authentification mais également pour la signature des requêtes et de réponses pour les échanges de zones de transferts. Le condensat est calculé sur l'ensemble du message DNS et certains champs complémentaires comme la date et l'heure de signature afin d'éviter les rejeux. Il faut impérativement que les systèmes communicants soient synchronisés car un décalage excédent cinq minutes implique le rejet du message. L'enregistrement TSIG est léger et n'apparaît ni dans les informations d'une zone, ni dans un cache. L'enregistrement TSIG est ajouté au message DNS. Le destinataire de ce dernier vérifie cet enregistrement, le supprime et traite les données utiles du message.

La méthode souffre cependant de quelques inconvénients qu'il est important de connaître. Les signatures, par exemple, ne tiennent pas compte de toute l'information offerte dans un fichier de zone, comme les données de délégation. Ce terme même de « signature » est d'ailleurs abusif puisqu'il n'y a pas de service de répudiation. Tous les détenteurs de la clé peuvent créer un message « signé » , y compris le destinataire. Ils permettent aussi de vérifier les enregistrements de manière individuelle et non l'intégralité du message de réponse de transfert de zone.

BIND a introduit TSIG à partir de sa version 8.2. Il faut s'assurer que la clé utilisée a une longueur minimale de 128 bits. La solution impose également qu'une clé ne serve que pour un couple de serveurs. Ainsi, chaque serveur secondaire échangeant avec celui primaire ne doit connaître qu'une clé unique.

La clé doit ensuite être manipulée avec toutes les précautions d'usage, en particulier pour la communiquer aux serveurs. Cette clé peut être produite par le biais d'utilitaires comme `dnssc-keygen`. Deux fichiers sont alors générés. Ils doivent être détruits ou accessibles au seul administrateur du système (*root*). Les informations ayant conduit à la création de la clé doivent disparaître des fichiers de configuration (nom de la clé ID, algorithme de signature, chaîne de caractères de la clé). La clé doit être déportée dans un fichier annexe placé dans un répertoire dédié. Chaque clé TSIG doit disposer d'un fichier propre.

### 4.4.2 *DNS Security Extensions* : DNSSEC

Comme il vient d'être vu, la méthode TSIG est en grande partie limitée par sa gestion et sa propagation de clés. Le partage de secrets est difficilement applicable à grande échelle, mais peut s'envisager entre quelques serveurs. Dans les autres cas, il faut envisager un chiffrement à clé publique.

DNSSEC a été créé pour aider les clients DNS (*resolvers*) à se protéger des données forgées de manière malveillante. Il ne protège pas les échanges de type transferts de zones entre serveurs. DNSSEC est une méthode décrite dans les documents RFC 2535, 4033, 4034 et 4035 afin de vérifier à chaque réponse l'intégrité des données et l'origine de celle-ci. DNSSEC s'appuie sur une architecture de clés publiques (`IGC/PKI`) et des algorithmes de chiffrement asymétrique pour construire une chaîne de confiance dans la vérification de signatures. Il faut néanmoins définir un serveur racine en tête de cette architecture. Pour utiliser DNSSEC de nouveaux enregistrements RR ont été définis, par exemple dans le standard RFC 2535, on note :

- DNSKEY : cette ressource porte la clé publique d'une zone (clé nulle - ou NULL KEY - autorisée) ;
- RRSIG : cette ressource enregistre la signature liée à la clé privée. Afin d'optimiser le temps, on signe un ensemble d'enregistrements individuels (de même nom, classe et type) appelés RRset.
- NSEC (anciennement NXT mais obsolète depuis le RFC 3755) : cette ressource permet de montrer qu'un nom n'existe pas et au contraire de fournir une réponse claire sur ce qui existe. Cela peut fournir beaucoup d'informations à une personne malveillante qui peut, par requêtes successives, reconstruire tous les noms d'une zone.
- DS, ou délégation de signature : l'enregistrement DS annonce la clé publique à utiliser pour vérifier les données d'une zone. Cette ressource est insérée par une autorité supérieure qui l'aura elle-même signé par sa propre clé privée afin de construire une chaîne de confiance.

Les aspects de confidentialité ne sont pas réellement pris en compte avec DNSSEC. DNSSEC ne protège également pas directement contre les attaques en déni-de-service ni contre l'usurpation d'adresse IP. Sa robustesse

dépend du référencement des clés par les serveurs parents qui peuvent également se faire compromettre et peut avoir un impact non négligeable sur les ressources du serveur.

Son utilisation se limite essentiellement à la construction pour son architecture DNS d'îlots sécurisés. A la date de rédaction de cette note d'information, DNSSEC est mis en œuvre par quatre gestionnaires de TLD uniquement : `.SE`, `.BR`, `.BG` et `.PR`. Des alternatives moins contraignantes ont également vu le jour :

- `DNSSEC Lookaside Validation` (DLV) proposé par l'ISC (*Internet Systems Consortium*) pour utiliser DNSSEC avec un système d'authentification tiers (abstraction de la vérification par hiérarchie donc) ;
- `Interim Trust Anchor Repository` (ITAR) proposé par la communauté RIPE. Cette solution fait abstraction d'une signature racine et réutilise les enregistrements de type DS de DNSSEC.

Des informations concernant l'état d'avancement de DNSSEC sont disponibles aux adresses suivantes :

- ICANN, "DNSSEC - Where we stand", juin 2008 :
  https://par.icann.org/files/paris/DNSSECRLamb_24Jun08.pdf
- ICANN, "DNSSEC at ICANN - Signing the root zone: A way forward toward operational readiness", 15 juillet 2008 :
  http://www.icann.org/en/announcements/dnssec-paper-15jul08-en.pdf
  https://ns.iana.org/dnssec/root.zone.signed
  https://ns.iana.org/dnssec/status.html
- Présentation par le CERT-In pour une réunion ICANN, "DNSSEC: A vision", février 2008 :
  http://delhi.icann.org/files/presentation-sagar-11feb08.pdf
- Portail d'information sur DNSSEC :
  http://www.dnssec.net
- RIPE NCC DNSSEC Policy, octobre 2005 :
  http://www.ripe.net/docs/ripe-359.html
- Standard RFC 4033, "DNS Security Introduction and Requirements", mars 2005 :
  http://www.ietf.org/rfc/rfc4033.txt
- Standard RFC 4034, "Resource Records for the DNS Security Extensions", mars 2005 :
  http://www.ietf.org/rfc/rfc4034.txt
- Standard RFC 4035, "Protocol Modifications for the DNS Security Extensions", mars 2005 :
  http://www.ietf.org/rfc/rfc4035.txt
- O. Kolkman, "DNSSEC HOWTO, a tutorial in disguise", mars 2007 :
  http://www.nlnetlabs.nl/dnssec_howto/
- S. Bortzmeyer, "RFC 4035: Protocol Modifications for the DNS Security Extensions", mars 2005 :
  http://www.bortzmeyer.org/4035.html

## 4.5 Anticiper le pire

D'une manière générale, il faut anticiper tout type de panne : défaillances matérielles, coupures d'électricité, bogues dans les logiciels, erreurs humaines, etc.

En cas de coupure de courant, une étape délicate peut être le redémarrage des machines. Le serveur DNS n'est fonctionnel qu'après son propre redémarrage. Les autres machines doivent dépendre le moins possible du DNS pour l'étape de démarrage. Il est donc préférable de placer dans des fichiers locaux de configuration les adresses IP plutôt que les noms, ou compléter les informations locales de nommage (/etc/hosts ou /etc/defaultrouter sous Linux).

Des mesures temporaires peuvent être envisagées selon les cas :

- utilisation d'un fichier /etc/hosts temporaire ;
- transformation d'un serveur esclave en maître pour une zone ;
- création temporaire de réponses au nom des serveurs racines (autorité sur la zone racine « . ») ;
- élimination de serveurs défectueux ou retournant des erreurs (option générale *blackhole* sous BIND qui précise une liste d'adresses pour lesquelles le serveur ne fournira pas de réponse).

## 4.6 Sinkhole et botnets

Des réseaux de machines zombies, ou *botnet* ont régulièrement recours aux services de gestion DNS pour des adressages dynamiques. Parmi les services les plus courants, il y a `dyndns.org` ou `no-ip.com`.

Les machines compromises des utilisateurs peuvent changer régulièrement d'adresse, celle-ci étant souvent attribuée par DHCP (*Dynamic Host Configuration Protocol*) par le fournisseur d'accès. Ces services permettent ainsi de changer très fréquemment les adresses IP associées à des noms de domaine malveillants. Quand une machine est compromise, elle peut par exemple chercher à joindre une machine de contrôle et donc se connecter à un canal IRC. Ce dernier utilise souvent ce principe de résolution dynamique.

Une méthode pour gêner les communications de ces machines zombies est de modifier les retours DNS pour des domaines réputés malveillants en les faisant pointer vers des ressources correspondant à un traitement spécifique (pots de miel, « trou noir », etc.). Cette modification peut être faite à grande échelle au niveau des fournisseurs d'accès afin d'avoir une vue générale des machines compromises participant au botnet.

Quand un contrôleur de bots subit une redirection DNS, il peut cependant effectuer une requête DNS sur son propre nom et réaliser que l'adresse IP retournée ne correspond pas à la sienne. Réciproquement, les bots peuvent effectuer la même opération et comparer la réponse DNS à une liste prédéfinie d'adresses qui pourraient servir à héberger un contrôleur. C'est alors un jeu de chat et de souris. Si l'un des contrôleurs semble avoir son nom détourné, les bots peuvent être mis à jour via un autre contrôleur, prévenir celui-ci de la découverte afin de faire un changement de nom. Un autre test pour le bot consiste aussi à vérifier avant toute communication importante que le contrôleur répond bien à un premier ensemble de commandes...

Ces solutions sont difficilement applicables de manière générale. Elles peuvent cependant être envisagées dans des cas particuliers et très ponctuellement.

- H.C. Jeong, KISA, "Botnet Handling with DNS Sinkhole", 2007 : http://www.cert.org/archive/pdf/BotSinkhole_KrCERTCC.pdf
- C.C. Zou, R. Cunningham, "Honeypot-aware advanced botnet construction and maintenance", DSN 2006 : http://www.cs.ucf.edu/ czou/research/honeypot-DSN06.pdf
- D. Dagon, C. Zou, W. Lee, "Modeling botnet propagation using time zones", NDSS 2006 : http://www.isoc.org/isoc/conferences/ndss/06/proceedings/papers/modeling_botnet_propagation.pdf

## 4.7 Différents serveurs, différentes précautions

### 4.7.1 Erreurs courantes

Nous revenons ici sur certaines erreurs assez fréquentes dans les configurations DNS.

Un nom de domaine complet doit se terminer par un « . » (racine de la hiérarchie DNS vue à la section 2.2). Dans les autres cas, il est relatif et sera le suffixe du domaine saisi. Par exemple :

```
FQDN :
        www.certa.ssi.gouv.fr.
```

Une erreur souvent constatée est d'oublier le point terminal à la fin d'un enregistrement de type « A » disposant déjà d'un nom complètement qualifié (`FQDN`). En particulier, si cet oubli concerne la zone *localhost* et l'enregistrement « A localhost », un serveur autorité pour la zone `MonDomaine.tld` répondra `127.0.0.1` si on le sollicite sur l'enregistrement :

```
localhost.MonDomaine.tld.
```

127.0.0.1 est l'adresse IP utilisée pour communiquer (en IP) avec sa propre machine : *localhost*, ou hôte local.

Il est ainsi possible de contourner la politique de sécurité ou de conduire des attaques de type « injection de code indirecte » (*cross-site scripting*).

Par exemple, sur une machine multi-utilisateurs de type `UNIX`, il est possible pour un utilisateur de lancer un serveur web sur un port non-privilégié (>1024), disons 9999, et d'inviter un autre utilisateur à visiter un serveur du type `http://localhost.MonDomaine.tld:9999`. La requête contiendra alors les en-têtes du fichier de session (*cookie*) décrit dans le standard RFC2109. Ce `cookie` est utilisé pour éviter l'usurpation de session. Or, dans ce cas précis, si un attaquant intercepte ces informations, il lui sera possible d'obtenir des informations d'identification de la victime.

Le point final (.) à la fin d'un enregistrement `FQDN` ou relatif au bouclage local `localhost` est indispensable et évite ce genre de désagréments. Il est important de bien vérifier que les enregistrements `FQDN` et *localhost* sont bien terminés par ces points finaux.

Plus généralement, les fichiers de zones doivent être produits de la manière la plus propre et la plus lisible possible. Sans cette rigueur, des erreurs peuvent très vite se produire, comme des entrées redondantes, des incohérences avec les fichiers de résolution inverse, etc.

Les enregistrements de type CNAME servent à enregistrer des alias. Leurs utilisations prêtent souvent à confusion.

```
www.exemple1.tld.          IN CNAME       www.exemple2.tld.
```

Il y a un enregistrement de type CNAME dont le nom est `www.exemple1.tld`. Le nom `www.exemple1.tld` est l'alias de `www.exemple2.tld`. Il n'est certainement pas, malgré l'abréviation de CNAME, un « nom canonique » (RFC 2181).

Un enregistrement CNAME peut, en principe, pointer vers un autre alias. Les standards n'interdisent pas de construire de cette manière un chaînage d'enregistrements CNAME. BIND ne détectera ni ne signalera un tel comportement. Le « client » resolveur peut lui éventuellement réagir par un message d'erreur. Une mauvaise configuration peut alors avoir des effets inattendus sur les serveurs : on peut imaginer le scénario où une boucle est créée, rendant la résolution impossible. Ce genre de scénario est envisageable dans le cas d'une compromission de cache ou d'une erreur humaine.

Le standard RFC 2181 ("Clarifications to the DNS Specification") rappelle à cet égard :

```
10.1. CNAME resource records

  The DNS CNAME ("canonical name") record exists to provide the
  canonical name associated with an alias name. There may be only
  one such canonical name for any one alias.
  ...
```

Les enregistrements CNAME peuvent également perturber les performances du serveur.

Il existe quelques contradictions en particulier sur les enregistrements CNAME associés à ceux MX, certains revendiquant des standards précisant le DNS (RFC 2181) et d'autres les protocoles de messagerie (RFC 2821).

Sans rentrer dans ce débat, dans le cas où un enregistrement MX pointe vers une ressource de type CNAME, le trafic des échanges DNS peut doubler. La première requête retournée ne sera qu'un CNAME : n'ayant pas de ressource additionnelle, (pas de RR additionnel), le client sera ainsi obligé d'émettre une seconde requête pour résoudre la ressource CNAME.

Par ailleurs, le fait d'utiliser une ressource de type CNAME complexifie la configuration et peut conduire à diverses erreurs de non-résolution.

Il est donc plus avisé pour un enregistrement MX de pointer directement vers une ressource de type A et pas vers une ressource de type CNAME. L'administrateur du serveur peut reprendre le « nom canonique » associé à l'alias de l'enregistrement CNAME et le copier directement dans l'enregistrement MX.

Pour débusquer ces erreurs courantes, plusieurs utilitaires sont fournis soit directement avec le logiciel offrant le service DNS, soit via des pages Web en ligne. Il est de manière générale fortement conseillé de vérifier avec grande précaution la configuration générale du serveur. Ces tests doivent être refaits à périodes de temps régulières.

- Politique de tests de ZoneCheck pour l'AFNIC :
  http://www.zonecheck.fr
  http://www.afnic.org/outils/zonecheck/tests
- Documentation Sendmail, section 9.3, "Set Up MX Records" :
  http://www.unix.com.ua/orelly/networking/sendmail/ch21_03.htm
- Domainmx.net, "Setting up MX Records in your DNS" :
  http://domainmx.net/mxsetup.shtml
- P. Balyoz, "DNS Utilities for Unix" (Domtools et Dlint) :
  http://www.domtools.com/

### 4.7.2 Configuration et maîtrise des paramètres liés au temps

La configuration des serveurs DNS est riche en paramètres de temps. Chacun a une signification propre : date d'expiration de RR, valeurs TTL, etc. Il faut choisir les valeurs avec précaution.

Un TTL de valeur nulle pour un RR s'interprète comme un enregistrement ponctuel pour la transaction en cours. Il ne devrait pas être mis en cache. C'est le cas par exemple des enregistrements SOA. Des TTLs très courts permettent une certaine flexibilité mais également peuvent engendrer davantage de trafic. Les serveurs distants ou

les resolveurs doivent interroger plus souvent les serveurs locaux. A l'inverse, des TTLs plus longs peuvent gêner les évolutions mais diminuent le volume de trafic et sont globalement plus robustes. De manière générale, il est intéressant d'augmenter le TTL si les informations changent peu. L'ordre de grandeur est alors d'une journée. La semaine est une valeur maximale rarement rencontrée. Au-delà, il devient très difficile de mettre à jour ou rectifier dans un temps raisonnable des informations erronées ou qui ont changé.

Les choix doivent se baser sur l'expérience car ils sont bien souvent affaire de compromis.

Les réponses négatives peuvent être mises en cache. Il faut alors limiter le TTL. Sous BIND, cela se fait avec la variable `max-ncache-ttl`. Par défaut, les anciennes versions (branche 8 du serveur) gèrent la mise en cache des réponses négatives en utilisant le champ de l'enregistrement SOA. La valeur est en général trop grande pour l'expiration d'une réponse négative. BIND dans sa version 9 permet aussi de limiter tout TTL sur les enregistrements placés en mémoire (`max-cache-ttl`). Cette pratique est intéressante pour mieux contrôler d'éventuels abus ou palier certaines mauvaises configurations d'autres serveurs. Il est possible de distinguer aussi le `lame TTL`. Cette durée représente le temps pendant lequel un serveur se rappelle qu'un serveur distant ne fait pas autorité sur une zone qui lui est déléguée. Une valeur suffisamment haute permet d'éviter au serveur de perdre son temps à interroger le serveur distant sur des données qui ne sont pas de son ressort.

Par exemple, il faut choisir la délai de rafraichissement d'une zone SOA en fonction de la fréquence des mises à jour prévues. Si cette zone est signée, cette valeur ne doit également pas être plus petite que la période de validité du RRSIG.

### 4.7.3 Incohérences entre primaires et secondaires

Il peut y avoir différentes raisons qui conduisent à des états incohérents entre les ressources retournées par les serveurs primaires et secondaires. Il peut s'agir d'un empoisonnement de cache ou d'un problème de configuration.

Par exemple, si le fichier de zone change plus fréquemment que les valeurs précisées dans les champs de rafraîchissement (Refresh et Retry) de l'enregistrement SOA, le serveur primaire risque de ne pas disposer des mêmes données que ses serveurs secondaires. Cette erreur est appelée *zone drift*. Les serveurs secondaires disposent de données qui ne sont plus d'actualité.

L'inverse consiste à définir des valeurs dans les champs de rafraîchissement trop courtes alors que cela n'est pas nécessaire. Par conséquent, les serveurs secondaires vont exiger des transferts de zone très fréquemment alors qu'ils ne sont *a fortiori* pas utiles. Cette erreur porte le nom de *zone trash*. Elle provoque une surcharge des serveurs et des liaisons et peut être exploitée pour des attaques de type déni de service.

Comment vérifier la cohérence des caches ?

Il n'y a pas de méthode absolue mais certaines propositions ont été faites. Le principe : récupérer périodiquement des copies de cache au niveau des serveurs récursifs et tester les nouvelles entrées en interrogeant directement les serveurs d'autorité ou d'autres serveurs récursifs. Toute anomalie est alors signalée. Cette méthode peut conduire à quelques faux positifs (cf. les architectures de distribution de contenus CDN).

- J. Avila, ONZRA, 2008 OARC DNS Operations Workshop, "Recursive DNS Cache Auditing" : https://www.dns-oarc.net/files/dnsops-2008/Avila-Recursive-cache-auditing.pdf
- Outil "CacheAudit" Sous licence BSD : http://www.onzra.com/CacheAudit-Latest.tgz

### 4.7.4 Résolution inverse

Le RFC 1918 définit les blocs d'adresses IP qui peuvent être utilisés pour des usages privés et internes. Ces adresses ne doivent donc pas paraître sur l'Internet et toute requête les concernant doit rester interne. Des serveurs comme SMTP ou HTTP sont régulièrement amenés à produire des résolutions inverses (requêtes PTR). Cela implique en particulier de configurer des zones d'autorité `in-addr.arpa` pour celles-ci au niveau des serveurs de noms locaux.

```
10.in-addr.arpa
168.192.in-addr.arpa
```

Une pratique courante consiste à ajouter les informations pour le réseau de bouclage (*loopback*) au serveur afin qu'il puisse communiquer avec lui-même. L'adresse du réseau est 127.0.0.0/24 et celle de la machine est 127.0.0.1. Chaque serveur doit être responsable pour lui-même de ce réseau. Le fonctionnement du serveur peut parfois être surprenant si cette entrée est manquante.

Il est donc possible de croiser des entrées de la forme :

```
localhost. IN A 127.0.0.1
```

Il ne faut cependant pas oublier le fameux « . » car cela signifierait alors que l'enregistrement n'est pas complètement qualifié (FQDN). En d'autres termes, des requêtes de la forme `localhost.DomaineQuelconque.tld` seraient résolues et permettraient de lancer des attaques par injection de code indirecte (XSS). Des codes de démonstration ont utilisé cette propriété pour une vulnérabilité du serveur d'impression CUPS.

http://www.certa.ssi.gouv.fr/site/CERTA-2008-ACT-012/

http://archive.cert.uni-stuttgart.de/bugtraq/2008/01/msg00270.html

### 4.7.5 Les résolutions par défaut, ou *Wildcard DNS Records*

Un enregistrement DNS dit *"wildcard"* dans une zone correspond à toute ressource qui n'existe pas.

Un exemple serait :

```
*.certa.ssi.gouv.fr. 3600 MX 10  machine.certa.ssi.gouv.fr.
```

Cette entrée permet de retourner pour toute requête vers un nom qui n'existe pas dans le domaine `certa.ssi.gouv.fr` un enregistrement MX pointant vers *machine.certa.ssi.gouv.fr*.

Cette fonctionnalité a été précisée dans les premières versions des standards DNS mais avec plusieurs zones d'ombre. Un standard a donc en juillet 2006 éclairci les choses :

– RFC 4592, "The Role of Wildcards in the Domain Name System", juillet 2006 :
http://tools.ietf.org/html/rfc4592

Quelles sont les motivations pour déployer une telle solution ?

– des bureaux d'enregistrement ou fournisseurs de noms de domaine peuvent être intéressés à rediriger les visiteurs commettant une erreur vers une page de publicité ;
– les fournisseurs d'accès peuvent rediriger des erreurs de frappes vers des pages de publicité (*catchcall typosquatting*).
– les gestionnaires de sites veulent que l'utilisateur aboutisse sur leur site quel que soit le préfixe utilisé (une modification du côté du serveur Web est alors aussi nécessaire) ;

Les mises en œuvre de cette solution divergent en fonction du serveur du nom utilisé.

Inversement, certains serveurs offrent la possibilité de filtrer les réponses utilisant de telles méthodes pour certains domaines. Des correctifs sont fournis par exemple pour BIND ou djbdns :

– Patch pour dnscache de djbdns :
http://tinydns.org/djbdns-1.05-ignoreip2.patch
– Modifications apportées dans BIND (`delegation-only`) :
http://www.isc.org/software/bind/documentation/arm95#id2567416
– AFNIC, "Avertissement pour les utilisateurs de BIND et des options `delegation-only`", juin 2007 :
http://www.afnic.fr/actu/nouvelles/general/NN20070611
– AFNIC, "L'AFNIC et les *wildcards* (jokers)", septembre 2003 :
http://www.afnic.fr/actu/nouvelles/international/NN20030918

Cette fonctionnalité pose problème et des incidents sont régulièrement relatés dans la presse. Par exemple, il y a des gestionnaires de ccTLD, comme le Cameroun qui l'utilise depuis août 2006. La proximité du `.cm` avec d'autres TLDs comme `.com` peuvent conduire à des détournements. D'autres TLDs ont opté pour une solution similaire, dont `.eu`. L'ICANN n'a pas d'autorité sur les décisions prises par les bureaux d'enregistrement nationaux.

Les erreurs de frappes de noms de domaine peuvent ainsi rediriger vers des pages publicitaires ou d'abonnement. Un chercheur a montré en 2008 qu'il était possible de s'attaquer directement aux sites Web qui sont pointés par ces directions. Ces pages sont souvent hébergées sur des serveurs tiers. La personne a donc directement ciblé la page par défaut sur un serveur de l'hébergeur en montrant qu'elle était vulnérable à une attaque par injection de code indirecte.

Ces redirections de sous-domaines posent également d'autres problèmes aux navigateurs qui définissent souvent des notions de confiance en fonction des domaines (accès aux fichiers de session, interprétation de codes mobiles, etc.).

– http://www.iab.org/documents/docs/2003-09-20-dns-wildcards.html
– Note d'information Microsoft, "LooseWildcarding" :
http://www.microsoft.com/technet/prodtechnol/windows2000serv/reskit/regentry/46758.mspx?mfr=true

- IAB, "IAB Commentary: Architectural Concerns on the use of DNS Wildcards", septembre 2003 : http://www.iab.org/documents/docs/2003-09-20-dns-wildcards.html
- Article Droits et Technonologie, "Typosquatting and the .eu Top Level Domain", juin 2007 : http://www.droit-technologie.org/actuality/details.asp?id=1049

#### 4.7.6 Recommandations pour *Berkeley Internet Name Domain* (BIND)

Plusieurs mesures de sécurité peuvent être appliquées pour consolider la configuration d'un serveur BIND. Certaines sont générales et applicables dans la majorité des cas, y compris pour d'autres services que le DNS. D'autres précautions dépendent ensuite de l'usage du serveur.

Parmi les premières mesures, il s'agit de ne pas publier la version du service. Ceci n'est pas une protection (principe de « sécurité par l'obscurité ») mais une bonne pratique. Un serveur peut répondre à l'enregistrement TXT associé au nom `version.bind`. Un test possible est :

```
$dig txt chaos version.bind.
```

"chaos" est une classe DNS (la plus utilisée étant "IN" pour Internet) associée à un ancien réseau, Chaosnet, mais qui est actuellement mise en œuvre par les versions récentes de BIND pour définir des zones d'information et de diagnostics comme `version`, `hostname` et `server-id`. Il suffit donc de refuser ce type de requêtes en modifiant dans la configuration (/etc/named.conf) :

```
  options {
  version none;
  };
```

L'option "none" n'existe cependant que depuis BIND 9.3.0.

Il faut également exécuter le service avec le moins de privilèges possibles.

Dans le cas où le serveur est utilisé pour répondre à des requêtes de l'Internet, il faut s'assurer qu'il n'accepte pas de requêtes récursives. La configuration doit donc avoir les lignes suivantes :

```
options {
recursion no;
};
```

Si le serveur est en charge de résoudre les noms de domaine pour un ensemble d'utilisateurs, alors il faut lui préciser les adresses IP autorisées à lui envoyer des requêtes récursives.

```
options {
allow-query {192.168.3/24; 192.168.4/24; };
#
allow-recursion { 192.168.3/24; 192.168.4/24; };
};
```

Cette liste doit être la plus précise possible.

BIND permet enfin de configurer plusieurs paramètres de manière assez fine :

- listes de contrôle d'accès ou ACL pour les transferts de zones (`allow-transfer`) ;
- limite du nombre total de transferts de zones simultanés demandés (`transfers-in`) ;
- limite du nombre total de transferts de zones simultanés servis (`transfers-out`) ;
- limite du nombre de transferts de zones par serveur NS (`transfers` ou `transfers-per-ns`) ;
- limite de la durée d'un transfert de zone (`max-transfer-time-in` ou `max-transfer-time-out`) ;
- limite des durées de rafraîchissement pour l'ensemble des zones gérées (`min-refresh-time` ou `max-refresh-time`) ;
- limite de la taille de la pile du processus *named* (`stacksize`) ;
- limite du nombre de clients servis simultanément, « clients » dans le sens d'entités envoyant des requêtes récursives (`recursive-clients`) ;
- limite du nombre de sessions TCP concurrentes (`tcp-clients`) ;
- limite du TTL sur les éléments placés en cache (`max-cache-ttl`) ;
- limite de la durée pendant laquelle le serveur se souvient qu'un serveur distant ne fait pas autorité sur une zone qui lui est déléguée (`lame-ttl`) ;
- etc.

Une bonne pratique avec BIND consiste également à séparer le serveur de résolution DNS du serveur de cache. Ils doivent si possible avoir des adresses IP distinctes. L'idée est, en cas de compromission, de ne pas donner à un attaquant les moyens de contrôler toutes les données (entrantes et sortantes). En cas de saturation d'un des deux services (DoS), l'autre pourra aussi rester opérationnel. Cela peut se faire de différentes façons, avec des machines physiques différentes, des cartes réseaux distinctes ou de plus simples alias IP.

Des mises en place possibles à partir d'une configuration BIND existante sont documentées aux adresses suivantes :

- D.J. Bernstein, "The importance of separating DNS caches from DNS servers" : http://cr.yp.to/djbdns/separation.html
- D.J. Bernstein, "How to run an external cache in place of an existing BIND cache, strategy 1" : http://cr.yp.to/djbdns/run-cache-bind-2.html
- D.J. Bernstein, "How to run an external cache in place of an existing BIND cache, strategy 2" : http://cr.yp.to/djbdns/run-cache-bind-2.html
- D.J. Bernstein, "How to run an external cache in place of an existing BIND cache, strategy 3" : http://cr.yp.to/djbdns/run-cache-bind-3.html

BIND dispose également d'un bon outil de débogage avec un peut moins de cent niveaux distincts, du moins verbeux (1) jusqu'au niveau de répartition des tâches (90). Le format n'est cependant pas toujours très lisible. Cet outil peut se lancer après le démarrage du service en envoyant des messages de contrôle au processus *named*. Par exemple, pour lancer puis arrêter un débogage au niveau 8 :

```
$rndc trace 8
$rndc notrace
```

En terme d'analyse, les versions BIND 8 et BIND 9 offrent la possibilité de récupérer la mémoire cache, via la commande :

```
Sous BIND 8
 $ndc dumpdb
Sous BIND 9
 $ rndc dumpdb -all
```

Cela crée dans le répertoire de travail un fichier `named_dump.db` contenant les données sur lesquelles le serveur fait autorité, les données stockées en cache et les indications initiales (serveurs root).

### 4.7.7 Microsoft DNS Server

Les serveurs de noms Microsoft actuels utilisent des mises à jour dynamiques pour les enregistrements. Ils supportent une variante de TSIG, GSS-TSIG (*Generic Security Service*). Les négociations de clés se font à l'aide d'un serveur Kerberos. L'administrateur n'a donc pas besoin de fournir manuellement la clé sur chacun des clients.

Attention cependant ; par défaut, les clients n'utilisent GSS-TSIG que si leurs mises à jour non signées sont refusées. Il faut faire une modification pour que les données signées soient envoyées en premier.

Microsoft a publié plusieurs articles dans sa base de connaissances afin de modifier certains comportements côté serveur et côté client :

- Base de connaissances Microsoft, KB 241352, « Comment faire pour empêcher la pollution du cache DNS » : http://support.microsoft.com/kb/241352/fr
- Base de connaissances Microsoft, KB 308919, "RFCs That Are Supported by Windows 2000 Domain Name System" : http://support.microsoft.com/kb/308919
- Base de connaissances Microsoft, KB 323380, « COMMENT FAIRE : Configuration de DNS pour l'accès à Internet dans Windows Server 2003 » : http://support.microsoft.com/kb/323380/fr
- Base de connaissances Microsoft, KB 318803, « Comment faire pour désactiver la mise en cache DNS côté client dans Windows XP et Windows Server 2003 » : http://support.microsoft.com/kb/318803/fr

– Base de connaissances Microsoft, KB 816592, « COMMENT FAIRE : Configuration de la mise à jour dynamique du DNS sous Windows 2003 » :
http://support.microsoft.com/kb/816592/fr
– Base de connaissances Microsoft, KB 315982, « COMMENT FAIRE : Configurer des enregistrements DNS pour votre site Web dans Windows 2000 » :
http://support.microsoft.com/kb/315982/fr
– Base de connaissances Microsoft, KB 246804, "How to enable or disable DNS updates in Windows 2000 and in Windows Server 2003" :
http://support.microsoft.com/kb/246804/fr
– Base de connaissances Microsoft, KB 330511, « Formation : Compréhension du service DNS et résolution des problèmes associés dans Windows 2000 » :
http://support.microsoft.com/kb/330511/fr

#### 4.7.8 DnsMasq - pdnsd - nscd - dnrd

Les systèmes Linux ne cachent généralement pas directement les réponses DNS. Des applications proposent donc d'offrir des services de mises en cache et de redirection. Parmi elles, il y a `DnsMasq` et `pdnsd`.

DnsMasq est un serveur de redirection DNS (*forwarder*) pouvant également faire office de serveur DHCP adapté aux réseaux de petites tailles. C'est une application relativement simple qui est fréquente dans les « box » Internet et autres routeurs ou points d'accès WiFi (OpenWRT par exemple). Il répond aux requêtes des noms de machines locales et retransmet éventuellement vers des serveurs DNS externes.

Il met en cache les enregistrements de type A, AAAA et PTR et supporte également les enregistrements MX et SRV.

Il supporte depuis la version 2.10 les connexions TCP. Celles-ci ne sont cependant pas mises en cache et elles ignorent les paramètres de configuration précisant les ports sources à utiliser. Si ces connexions ne sont pas utiles, il est donc important de bloquer ces ports.

DnsMasq permet de récupérer le contenu du cache, y compris les réponses négatives. Pour cela, il faut envoyer un signal SIGUSR1 (`kill USR1 pid`) et avoir lancé le processus avec l'option `-q` ou `-log-queries` en mode non-démon (`-no-daemon`). Le cache n'est cependant pas stocké sur le disque et ne survit pas au redémarrage de la machine.

http://www.drazzib.com/docs:admin:dnsmasq

L'utilitaire Proxy DNS Daemon, pdnsd, est très semblable à DnsMasq. Il supporte en revanche plus de types d'enregistrements et supporte localement les enregistrements SOA, A, PTR, NS, CNAME et MX. Ses paramètres de configuration associés au DNS sont plus nombreux également. Il est possible de désactiver les connexions TCP et de préciser une plage de ports source UDP à utiliser pour les requêtes (`query_port_start` et `query_port_end`).

Sa configuration peut être modifiée pendant que le service fonctionne via la commande `pdnsd-ctl`. Elle permet également d'interroger le cache et d'effacer des enregistrements dans celui-ci. pdnsd conserve le cache (*permanent caching*) en cas de redémarrage.

http://www.phys.uu.nl/ rombouts/pdnsd/index.html

L'utilitaire *Name service cache daemon*, nscd, est un utilitaire également très semblable. Il offre la possibilité de contrôler assez précisément les différentes mises en cache (issues de certaines interfaces de la bibliothèque C comme *gethostbyname* ou `getaddrinfo`), le TTL et les mises en cache négatives.

http://udrepper.livejournal.com/16362.html

L'utilitaire *Domain Name Relay Daemon*, DNRD, offre les mêmes services que les deux précédents. Il a également une option d'équilibrage (*balancing*) si la liste des serveurs de redirection contient au moins deux éléments.

#### 4.7.9 djbdns

djbdns est une solution développée par D.J. Bernstein comme alternative au complexe projet BIND. Il est constitué de différents composants ayant chacun un rôle bien déterminé : `tinydns` est le serveur DNS, `dnscache` le client et gestionnaire de cache, etc. Les fonctionnalités sont sous forme de programmes distincts, ce qui permet d'avoir un serveur de complexité modérée et dont les fonctionnalités sont plus facilement contrôlées. Un ensemble d'outils pour les clients est également fourni. Le code source est depuis décembre 2007 dans le domaine public.

Le format des journaux de djbdns est peu documenté. Les adresses IP sont affichées sous forme de 8 caractères hexadécimaux. Les autres informations récupérables sont : les numéros de ports, les identifiants de requêtes et le type de requête, sous forme de 4 caractères hexadécimaux. Les types les plus courants sont :

```
0001 A
0002 NS
0005 CNAME
0006 SOA
000c PTR
000f MX
0010 TXT
001c AAAA
0026 A6
00fb IXFR
00fc AXFR
```

Le format global de chaque ligne ressemble donc à :

```
#tinydns tente de répondre à une requête
ip:port:id + type_requête nom_domaine
# tinydns rejète une requête car il n'a pas d'autorité pour répondre
ip:port:id - type_requête nom_domaine
# tinydns reçoit une requête à laquelle il ne peut répondre (RCODE 4 - NOTIMP)
ip:port:id I type_requête nom_domaine
# tinydns reçoit une requête pour une classe différente de IN ou 255 (RCODE 1 FORMERR)
ip:port:id C type_requête nom_domaine
# tinydns reçoit une requête malformée ou vide
ip:port:0000 / 0000 .
```

Un script PERL a été publié pour rendre les journaux plus lisibles. Il n'est pas développé dans le cadre du projet mais est cité comme bon utilitaire par D. Bernstein :

http://www.hungry.com/ fn/dnscache-log.pl.txt

#### 4.7.10 PowerDNS

Le serveur de noms PowerDNS permet de filtrer depuis sa dernière version 3.1.7, de modifier les réponses DNS observées. Cela peut être exploité pour filtrer certains domaines malveillants (principe de listes noires ou blanches) ou faire des redirections de type *wildcard* comme détaillé dans le paragraphe 4.7.5.

PowerDNS a également une interface Web de gestion mais qui n'est pas active par défaut. Elle est accessible si les variables de `webserver` sont correctement renseignées dans le fichier de configuration (`pdns.conf`). L'accès à cette interface doit être sécurisé si elle est utilisée. La seule possibilité offerte par l'application est actuellement une authentification HTTP simple avec l'usage d'un mot de passe unique pour tous les utilisateurs.

- Section Securité de la documentation PowerDNS :
  http://doc.powerdns.com/security-policy.html
  http://doc.powerdns.com/monitoring.html
  http://doc.powerdns.com/considerations.html

# 5 Surveillance dédiée

## 5.1 Journalisation

Comme tout service qu'il faut sécuriser, il est important de surveiller le bon fonctionnement de son architecture DNS. Sans cette opération, il sera très difficile de détecter une malversation. Le prérequis est d'avoir un minimum de journaux et de traces à analyser. Ces derniers doivent être collectés de différentes sources, en particulier les serveurs DNS et des équipements réseau. Cela permet d'avoir différentes vues.

Il faut alors s'assurer que les flux observés sont en adéquation avec la politique de sécurité adoptée.

Certains réseaux de machines compromises (*botnet*) effectuent un nombre élevé de requêtes DNS. Il peut alors être judicieux de journaliser efficacement toute requête au niveau du serveur DNS, afin d'analyser les fréquences

de requêtes de ressources RR : `A`, `PTR` et `MX`. Bind 9 offre la journalisation `querylog` qui permet de récupérer les requêtes au format suivant :

```
Mois jour h:m:s hostname named[PID]: client IP address#port: query: contenu de la requêt
Dec 26 15:00:05 labo named[344]: client 192.168.0.4#40010: query www.certa.ssi.gouv.fr I
```

Des statistiques peuvent être régulièrement consultées, comme :

- le nombre d'adresses IPv4 sources effectuant les requêtes DNS ;
- la distribution des types et des contenus des requêtes DNS ;
- la distribution des contenus de requêtes DNS provenant de l'extérieur du réseau.

Les erreurs de résolution de noms se manifestent par une réponse DNS dite `NXDOMAIN` (champ RCODE de la trame de valeur 3, "*Name Error*). Elles présentent essentiellement de l'intérêt quand elles sont émises par des serveurs de nom d'autorité car elles signifient alors que le nom demandé dans la requête n'existe pas. La surveillance de ces trames particulières ainsi que leur volume peut être un bon indicatif de dysfonctionnement ou d'anomalie.

Pour les plus assidus, il est possible, par quelques formules mathématiques d'entropie et de seuils d'établir des niveaux d'alertes.

## 5.2 Détection et analyse

En fonction des moyens disponibles, il peut s'avérer très utile d'établir une politique de surveillance adaptée pour le DNS.

Il existe des outils d'analyse passifs, dits de réplication. le principe est le suivant : un outil capture les trames réseau DNS et essaie de reconstruire des répliques des zones, basées sur les réponses des serveurs. Cette technique permet d'identifier des problèmes de configuration (résolution inverse), ou des activités malveillantes (botnet cherchant à se connecter vers d'autres serveurs DNS par exemple).

Les commandes de base comme `nslookup` ou `dig` restent aussi des outils d'analyse essentiels.

```
Pour vérifier le fichier des serveurs racines :
(ftp://ftp.internic.net/domain/named.root)
$dig +nocmd . NS +noall +answer +additional

Pour tracer le << chemin >> de résolution :
dig certa.ssi.gouv.fr +trace

Les informations (SOA) envoyées par les serveurs :
dig certa.ssi.gouv.fr +nssearch
dig SOA certa.ssi.gouv.fr
```

Le mot clé « ANY » correspond au type de requête (QTYPE selon le RFC 1035) « * ». Cette requête demande tous les enregistrements au serveur. Cependant le RFC ne précise pas comment les serveurs doivent se comporter en fonction de cette requête (en particulier les caches). Ce dernier n'est cependant pas obligé de répondre. Il peut se contenter de retourner les enregistrements de type NS. Il est donc préférable d'émettre des requêtes plus précises pour mieux interpréter les réponses.

- P. Heinlein, "DIG HOWTO", mai 2006 :
  http://www.madboa.com/geek/dig/

# 6 Conclusion

Le lecteur aura ici bien compris que ce document ne fournit qu'un aperçu des problématiques associées au DNS, ce sujet ne pouvant être couvert simplement et en quelques pages. Des risques existent et restent menaçants sur les architectures DNS. Le fait que ce protocole soit maintenant devenu parmi les plus « anciens » ne doit pas faire oublier certaines bonnes pratiques. Son apparente simplicité est trompeuse. Le déploiement et la maintenance de cette architecture doivent prendre toute sa part dans la politique de sécurité.

# 7 Documentation complémentaire

## Références


[AFNI1] Supports de cours de l'AFNIC :
http://www.afnic.fr/doc/formations/supports

[AFNI2] Module d'autoformation sur le DNS par l'AFNIC :
http://www.afnic.fr/doc/formations/autoformation/dns

[CERTA] Bulletin d'actualité CERTA-2007-ACT-018 du 04 mai 2007 :
http://www.certa.ssi.gouv.fr/site/CERTA-2007-ACT-018.pdf

[BORTZ] Bloc-notes de S. Bortzmeyer avec de nombreuses ressources en français :
http://www.bortzmeyer.org

[CYMR] Page de l'équipe Team Cymru, indiquant l'état relatif des différents serveurs racines et gTLDs :
http://www.cymru.com/monitoring/dnssumm/index.html

[DITL8] "DITL 2008 Analysis", S. Castro, Cooperative Association for Internet Data Analysis CAIDA, juin 2008 :
http://www.caida.org/publications/presentations/2008/oarc_castro_ditlanalysis/

[DJBD] Site de DJBDNS développé par Dan Bernstein :
http://cr.yp.to/djbdns.html

[DNSITR] Outils de surveillance et travaux de recherche lancés par CAIDA, « dns-itr » :
http://www.caida.org/research/dns/dns-itr/

[DNSLOG] Outil d'analyse de trafic DNS et de réplication passive :
http://www.enyo.de/fw/software/dnslogger

[DNSSEC] Documentations sur le thème "DNS Threats and DNS Weaknesses (DNSSEC - DNS Security Extensions)" :
http://www.dnssec.net/dns-threats

[GTLD] Site de l'ICANN concernant la gestion des gTLDs :
http://gnso.icann.org

[IETF] A. Durand, J. Ihren, P. Savola. "Operational Considerations and Issues with IPv6 DNS", IETF Draft :
http://www3.ietf.org/proceedings/06mar/IDs/draft-ietf-dnsop-ipv6-dns-issues-12.txt

[INTIV] INT Evry : « Le tour du Net en questions : DNS » :
http://www-public.int-evry.fr/ maigron/Internet/DNS.html

[ISOC] FAQ concernant les serveurs racines :
http://www.isoc.org/briefings/020/

[NIST] Guide de mise en œuvre du DNS, par le NIST, « Secure Domain Name System (DNS) Deployment Guide » (anglais) :
http://csrc.nist.gov/publications/nistpubs/800-81/SP800-81.pdf

[RFC1034] RFC 1034 « DOMAIN NAMES - CONCEPTS AND FACILITIES », novembre 1987 :
http://rfc.net/rfc1034.html

[RFC1035] RFC 1035 « DOMAIN NAMES - INPLEMENTATION AND SPECIFICATION », novembre 1987 :
http://rfc.net/rfc1035.html

[RFC1912] RFC 1912 « Common DNS Operationnal and Configuration Errors», février 1996 :
http://rfc.net/rfc1912.html

[RFC3833] RFC 3833 « Threat Analysis of the Domain Name System (DNS) », August 2004 :
http://www.ietf.org/rfc/rfc3833.txt

[RSERV] Site des serveurs DNS racines :
http://root-servers.org/


## Gestion détaillée du document

**25 juillet 2008** version initiale.