Réseau avancé

**DNS**

**Intro**

TCP/IP utilise des adresses sur 32 bits

DNS = Domain Name System = ensemble de protocoles et services qui permettent aux users d'utiliser un nom symbolique hiérarchisé pour un ordinateur.

Fonction la plus connue: faire correspondre un nom et une IP.

Plus facile pour les gens de retenir le nom que l'adresse IP

Le nom peut rester le même mais l'admin réseau peut changer l'IP associée

Avant le DNS, l'utilisation des noms se faisait par l'usage de fichiers HOSTS (contient des IP avec des noms associés). Sur Internet, le fichier était administré de façon centrale et on devait donc charger une copie de ces fichiers réguliérement. Avec l'augmentation de machines sur Internet, il a fallu trouver une autre solution => DNS.

Le DNS se situe sur la couche 7 du modèle OSI, on peut donc l'utiliser en UDP ou en TCP, il fonctionne sur le port 53

**2 Structure du DNS**

Le DNS, travaille en divisant Internet en ensemble de domaines ou réseaux qui peuvent être divisés en sous domaines.

La structure ressemble à une arborescence où le sommet et le Top Level Domain. Ce niveau est géré par L'Internet Network Information Center = Internic. Il comprend les domaines pour organisations (org, com, info, edu, etc) et un domaine par pays. Les domaines de pays sont un groupe de deux à trois lettres (be, fr, uk, etc).

Un domaine peut contenir soit des hosts, soit des sous domaines. Chaque domaine s'identifie par rapport à son parent en étant séparé par un point. on parle souvent pour les noms de domaines de noms pointés.

E.G. : Le site de la HELHa est un sous domaine du domaine be = helha.be . Chaque sous domaine peut dont aussi contenir un sous domaine ou des hosts. On peut donc retrouver des ordinateurs tels que www.helha.be ou ac.helha.be (sous domaine de l'extranet ac.helha.be)

**3 Serveur de noms**

Chaque serveur de noms de domaines gère une zone distincte du réseau. L'ensemble des machines gérées par le serveur est une zone. Un serveur peut gérer plusieurs zones.

On peut donc imaginer le domaine helha.be comprenant la zone helha.be et la zone info3.helha.be. Ces deux zones peuvent être gérées par un serveur unique ou deux serveurs distincts. Ces serveurs ont donc l'autorité sur leur zone.

La plupart des zones possèdent plusieurs serveurs de noms:

- Un serveur de noms primaire

- Un serveur de nom secondaire ou de sauvegarde.

Un serveur primaire reçoit les données pour sa zone de fichiers locaux, tout changement dans une zone se fait sur le serveur de noms primaire.

Un serveur secondaire reçoit les données pour sa zone d'un autre serveur de noms qui a l'autorité sur cette zone. Ces serveurs communiquent ensemble par un protocole de transfert de zone.

DNS apparait comme un ensemble de zones emboîtées, chaque serveur de noms communique avec un autre serveur de noms au dessus de lui et des fois aussi en dessous de lui s'il y'en a un.

Chaque zone a au moins un serveur responsable qui connait la correspondance entre les noms d'ordis et les adresses dans sa zone.

Chaque serveur connait aussi l'adresse d'au moins un autre serveur de noms.

Les messages qui transitent entre les serveurs de noms utilisent le protocole UDP car elle apporte de meilleures performances MAIS TCP est utilisé pour la mise à jour des DB grâce à sa fiabilité

Quand un user a besoin de résoudre un nom vers une IP, une demande est envoyé vers un service de résolution. Celui-ci envoie une demande vers un serveur de nom qui va voir s'il contient l'adresse correspondante dans ses tables. S'il ne la trouve pas, il envoie donc la demande vers un autre serveur. Les serveurs de noms et les resolvers utilisent une cahce locale pour mémoriser le résultat des requêtes les plus récentes à l'extérieur de la zone.

**4 Résolution des noms**

Il y'a deux types de demandes qu'un client peut faire à un serveur DNS: récursif, itératif et inverse.

Résolution de nom: Un serveur DNS peut être le cient d'un autre serveur DNS

Le serveur DNS reçoit une demande du client. Ce système envoie une requête à un serveur différent pour chaque domaine.

Une requête récursive place le moins possible de charge sur le client (tout est sur le serveur original). Le serveur DNS recevant la requête peut retourner une réponse avec l'IP ou un message d'erreur expliquant pourquoi il ne peut pas le résoudre.

La plupart des clients DNS utilise des requêtes récursives mais il y'a les requêtes itératives (qui nécessitent un plus grand effort de la part du client).

Quand un serveur DNS reçoit une requête itérative, il ne retourne que l'information qu'il possède sans effectuer de requête suppléntaire. Si le sevreur ne possède pas d'information, il peut fournir l'adresse d'un sevreur de noms racine.

Le client est responsable de la transmission de sa requête à un des serveurs indiqués dans la réponse. Ces requêtes ne sont pas réservés aux clients. Quand un serveur DNS envoie une requête à un autre serveur DNS, il fonctione aussi comme un client et peut être configurer pour envoyer un des types de requêtes

La première requête est une récursive envoyé par le résolveur vers le DNS paramétré dans la pile TCPIP. S'il ne peut pas répondre, il va envoyer différentes requêtes pour avoir la réponse => requêtes itératives.

Les serveurs DNS sont souvent configurés pour accepter les requêtes itératives pour leurs zones mais seulement les récursives pour leur client

Pour améliorer les performances, les résolveurs et serveurs récursifs possèdent un cache interne. Dans un réseau local, on peut utiliser un serveur qui n'a pas d'autorité sur les zones mais qui sert de serveur de cache en relayant les demandes récursives vers un autre serveur externe.

Si les serveurs du TLD fournissent des informations sur les serveurs de noms capables de répondre aux requêtes itératives. Le serveur récursif ne peut pas déterminer de façon dynamique la liste des adresses des serveurs du TLD. Ces infos sont donc dans tous les serveurs DNS. La première démarche est de mettre à jour cette liste d'adresses dès l'installation du serveur DNS

Il y'a 13 serveurs racine et appartiennent au même domaine root-servers.net . Les noms de ces serveurs vont de a.root-serves.net jusqu'à m.root-servers.net . Ils sont gérés par douze organisations : 2 euro, 1 japonaise et 9 US. Gestion multicast => répartis à travers le monde.

**5 Les fichiers DNS**

On se limitera à la config d'un serveur DNS BIND (9). Permet d'appréhender les RFC que la config trop limitative d'un serveur Microsoft.

On retrouve dans le répertoire /etc/bind, les fichiers:

named.conf = fichier principal comprenant des inclusions des deux autres fichiers, comprend aussi la dft des zones locales

named.conf.local = correspond au fichier de dft des différentes zones qu'on souhaite ajouter au serveur

named.conf.options = options de config globale du serv et valables par défaut pour toutes les zones définies.

Une des options correspond à l'emplacement ou vont se trouver les fihciers associés à chaque zone définie. le nom est laissé à la liberté de l'admin du serveur.

e.g.: ```

options {

directory "var/cache/bind";

}

```

Cette option précise que les fichiers de zone se situeront dans le répertoire donné ci-dessus.

5.1 Définition des zones (named.conf.local)

Dans named.conf, on peut voir l'instruction -> include "/etc/bind/named.conf.local";

On retrouve la définition des zones

5.2 Types de zones

Il y'a cinq types de zones différentes:

master : Serveur a une copie maître des data pour la zone + a autorité sur la zone

slave : Réplique d'une zone maître. Contient la liste "masters" et contient une ou plusieurs IP auxquels l'esclave peut se co pour obtenir une update des fichiers

stub : Comme une zone esclave MAIS contient seulement les records NS de la zone maitre, pas la zone entière

forward : Utile pour rediriger vers un autre serveur. Est dans la dft des options mais du coup vient cacher les déclarations similaires dans les options.

hint : ensemble des serveurs de noms racines.

**5.3 Exemples de dft de zones**

```

zone "helho.be" in {

type master;

file "helho.be.zone";

};

```

> Le serveur a autorité sur la zone helho.be -> type master. (Nom du fichier = helho.be.zone sera dans le directory indiqué par l'option dans named.conf.options)

```

zone "helha.be" in {

type slave;

file "slave.helha.be.zone";

masters{

193.190.66.4;

};

};

```

> Le serveur n'a pas d'autorité. Sert de serveur de sauvegarde -> type slave. (Nom du fichier = slave.helha.be.zone) + contenu pas défini sur le serv -> reçoit une copie du serveur qui a autorité (IP fournie dans `masters`)

DANS NAMED.CONF :

```

zone "."{

type hint;

file "/etc/bind/db.root";

};

```

> fichier contenant la liste des serv du TLD.

**5.4 Fichier de zone (Association directe)**

Fichier qui contient des enregistrements de ressources (records) pour la zone gérée de ce serveur. En format ASCII (facile à m-à-j).

Le premier record est toujours un record SOA :

```

IN SOA <source host> <email de contact><serial number><refresh time><try interval><expire time><TTL>

```

source host = hote où le fichier est maintenu

email = email du responsable du fichier de zone pour le domaine

serial no = Numéro de version de fichier (incrémenté à chaque modif). Utilisé pour voir s'il on doit m-à-j

refresh rate = temps en seconde qu'un serv secondaire doit attendre pour savoir si le fichier de zone a été modif et qu'il faut m-à-j.

try int = Intervalle en seconde qu'un serv secondaire doit attendre pour réessayer un transfert raté.

expire time = temps en seconde qu'un serv secondaire doit attendre pour essayer de charger une zone

TTL = Time To Live = Temps associé à la mise en cache sur les serveurs récursifs, valeur de cache négative

Après le SOA, nous pouvons avoir des records:

A = enregistrement IPv4 => correspondance nom - IP

AAAA = enregistrement IPv6 => correspondance nom - IP

CNAME = enregistrement de nom canonique = crée un alias vers un nom d'hote => Utile quand les noms de domaines multiples sont résolus vers une même IP

MX = enregistrement échange de courriel. Associé à une préf quand pls serveurs sont définis pour le même domaine.

NS = enregistrement de serveur de nom dans les serveurs DNS. Important d'y renseigner le serv primaire et les serv de backup pour que les notifs soient envoyés quand le serial no du fichier de zone change et que le serveur a restart.

TXT = Insère un texte quelconque. Informations de type SPF (Sender Policy Framework) pour valider les serveurs de messagerie pour un domaine donné. Utilisé contre le spam.

EXEMPLE:

```

$TTL 1W

@ IN SOA ns.helho.be. root.helho.be. (

92 ;serial

2D ;refresh

4H ;retry

6W ;expiry

20W) ;minimum

@ IN NS ns.helho.be.

@ IN NS ns.belnet.be.

@ IN MX 1 barracuda.helho.be

@ IN MX 2 mail.helho.be

@ IN TXT "v=spfl mx ptr ip4:193.191.131.9 -all"

mail IN TXT "v=spfl a -all"

barracuda IN TXT "v=spfl a -all"

ns IN A 193.191.131.2

ns1 IN A 193.191.131.10

www IN A 193.191.131.8

```

**5.5 Fichier de zone (Association inverse)**

Fct principale d'un DNS est de retrouver IP à-p-d son nom.

Pour limiter l'accès à certains services (pour les membres du domaine), un domaine "in-addr.arpa" a été créé dans l'espace DNS. Nom complet comprend la partie d'IP associée au no du r;eseau mais dans le sens inverse, suivi du domaine in-addr.arpa

EXEMPLE: 193.191.131.0 => 3 premiers octets sont ceux du réseau. Nom de la zone inversée = 131.191.193.in-addr.arpa

>Pour une IP classless PLUS DUR.

EXEMPLE: 193.191.131.0/27 comprend les 3ers octets mais les 3 bits de poids fort de l'octet suivant => besoin que le serveur qui a autorité sur 193.191.131.0/24 délègue la responsabilité de la reso inverse vers le serveur ns.helho.be

=> Provider utilisera dans son fichier de zone inverse (193.191.193.in-addr.arpa) et qui comprendra :

0/27 NS ns.helho.be

1 CNAME 1.helho.131.191.193.in-addr.arpa.

> Sur le serv helho.be on doit donc avoir une zone inverse helho.131.191.193.in-addr.arpa.

Contenu associé comprendra un record SOA suivi des records NS + PTR

EXEMPLE:

```

$TTL 1W

@ IN SOA ns.helho.be root.helho.be (

51 ; serial

2D ; refresh

4H ; retry

6W ; expiry

20M) ; minimum

@ IN NS ns.helho.be.

@ IN NS ns2.helho.be.

2 IN PTR ns.helho.be.

3 IN PTR nathalie.helho.be.

4 IN PTR kameleon.helho.be.

5 IN PTR noname.helho.be.

```

> Si on veut tester le fonctionnement de notre DNS on peut utiliser sous linux : dig -x 193.191.131.10

On obtient alors:

```

;;QUESTION SECTION:

;10.131.191.193.in-addr.arpa. IN PTR

;;ANSWER SECTION:

10.131.191.193.in-addr.arpa. 86321 IN CNAME 10.helho.131.191.193.in-addr.arpa.

10.helho.131.191.193.in-addr.arpa. 172800 IN PTR acolad.helho.be.

;;AUTHORITY SECTION:

helho.131.191.193.in-addr.arpa. 172800 IN NS ns2.helho.be.

helho.131.191.193.in-addr.arpa. 172800 IN NS ns.helho.be.

;;ADDITIONAL SECTION:

ns.helho.be. 172800 IN A 193.191.131.2

```

Première réponse = record CNAME mis en place par le provider

Suite = réponse PTR du serveur ns.helho.be.

**5.6 Zone de type hint et fichier associé**

fichier de cache contient les infos nécessaires pour résoudre les noms à l'extérieur du domaine qui a autorité. Contient les noms et adresses des serveurs racine. Peut être chargé à l'adresse : FTP://rs.internet.net/domain/named.cache et permet l'update du fichier associé au serveur DNS

**5.7 Record MX dans le DNS**

1) Courier envoyé de emile@skynet.be à marcel@helho.be , le courriel arriver sur le serveur de mail sortant configuré dans le client, le serveur vérifie s'il gère le domaine, si oui le courrier est placé dans la boite de réception.

2) Le destinataire au domaine helho.be , le serv de courrier sortant va interroger le serveur DNS qui a autorité sur la zone pour connaitre le reco-rd MX.

3) Quand il a reçu l'info sur le serveur MX, le serveur de mail sortant peut entrer en contact avec le serveur de mail du destinataire.

4) Le serveur mail.helho.be vérifie que le destinataire est géré par lui et est placé dans la boite de réception correspondante.

**DNSSEC**

Pour assurer l'authenticité et l'intégrité, DNSSEC se base sur un modèle de crypto à clé publique. Un serveur d'autorité calcule un hash puis le signe avec une clé privée avant d'envoyer le paquet avec le hash au résolver, qui pourra vérifier l'authenticité et l'intégrité des données en déchiffrant et associant la clé publique à la clé privée. Permet de garantir le fonctionnement des demandes de résolutions classiques => records DNS avec des enregistrements DNSKEY, RRSIG, NSEC, DS ET NSEC3, NSEC3PARAM ajoutés.

**Enregistrements liés aux signatures**

Soit la zone esit:

```

$TTL 86400

$origin esit.be.

@ 10D IN SOA ns1.esit.be. esit-it.experts.be. (

2015010103 ; serial

5H ; refresh

3600 ; retry

1W ; expire

1H ; TTL negatif

)

@ IN NS ns1.esit.be.

@ IN NS ns1.belnet.be.

@ IN NS ns2.belnet.be.

@ IN A 193.191.131.25

@ IN AAAA 2001:06a8:3480:2::26

ns1 IN A 193.191.131.25

ns1 IN AAAA 2001:06a8:3480:2::25

www IN A 193.191.131.25

www IN AAAA 2001:06a8:3480:2::26

```

> Enregistrements regroupés en fct du nom et type pour former des RRSET (Record Resource SET).

DNSSEC se porte sur un RRSET et pas sur un enregistrement individuel => On prend donc les enregistrements NS associés à esit.be

```

@ IN NS ns1.esit.be.

@ IN NS ns1.belnet.be.

@ IN NS ns2.belnet.be.

```

**Enregistrements DNSKEY**

Utilisé pour transmettre une clé publique entre le résolveur et le serveur. Associée à la clé privée utilisée par le serveur pour signer les hash des records RRSET. Le resolver utilise donc la clé publique écrit dans DNSKEY pour vérifier la signature du serveur.

Une clé publique est associée à une zone, pas à un serveur.

Deux types de clefs : KSK et ZSK.

KSK = signe seulement les DNSKEY RRSET

ZSK = signe les autres RRSET dans la zone

Structure des records dans la zone signee esit.be

```

@ 86400 DNSKEY 256 3 8 (

<clé publique>

); key id = 6228

```

86400 = TTL

256 = champs flags (256 = ZSK , 257 = KSK)

3 = protocole -> tout le temps 3 pour un record DNSKEY

8 = champ algorithme. valeurs pouvant etre utilisés :

0 = reserved

1 = RSA/MD5

2 = Diffie/Hellman

3 = DSA/SHA-1

4 = reserved

5 = RSA/SHA-1

8 = RSA/SHA-256 |

10 = RSA/SHA-512 | recommandé

**Record RRSIG**

Contient la signature de l'enregistrement envoyé par le serveur. Signature obtenue en signant le hash du RRSET avec la clé privée du serveur (contenue dans le record DNSKEY). Signature vérifiée par le resolver

=> IL EXISTE UN RECORD RRSIG POUR CHAQUE RECORD DE LA ZONE DANS LE FICHIER DE ZONE SIGNÉ

```

86400 A 193.191.131.25

86400 RRSIG A 8 2 86400 20140714102050 (

20140614092050 esit.be.

<clé > )

```

A = 'type covered' = type d'enregistrement signé => RRSET de A

8 = algo utilisé (RSA/SHA-256)

2 = champ label => niveau de hiérarchie dans le nom contenu dans le RRSET. Ici esit.be contient deux champs (be et esit) => 2.

Dans la même zone, pour ns1.esit.be => 3 (be esit ns1)

86400 = TTL d'origine

20140714102050 = date d'expiration de la signature

20140614092050 = date de signature

6228 = ID de la clé utilisé. Clé dans le record DNSKEY

esit.be = zone qui détient la clé

**Les enregistrements liés à la preuve de non existence.**

Intro

On peut assurer l'origine et l'intégrité des réponses si l'on reçoit une réponse.

Que se passe-t-il quand on envoie une requête au serveur ?

> Avec le DNS ? Aucune idée MAIS il y'a le NSEC et NSEC3

NSEC

Contient les infos servant à identifier les records qui existe pour un nom donné + autres noms qui existent dans la zone. Sert à voir si un record ou un domaine existe (ou pas).

EXEMPLE:

```

ns1.esit.be. 3600 NSEC www.esit.be. A AAAA RRSIG NSEC

```

www.esit.be. = nom du domaine suivant

A AAAA RRSIG NSEC = liste de RSET qui correspond au domaine suivant

NSEC = dernier nom de domaine qui pointe vers le premier (dans la liste).

Problème = on peut récupérer l'ensemble des noms de domaine sans que les transferts soient validés pour une zone. Nous pouvons utiliser ldns-walk dans linux pour montrer cette faille.

EXEMPLE:

```

~ldns-walk @127.0.0.1 esit.be

esit.be. esit.be. A NS SOA AAAA RRSIG NSEC DNSKEY

ns1.esit.be. A AAAA RRSIG NSEC

www.esit.be. A AAAA RRSIG NSEC

```

> Pour éviter celà le NSEC3 et NSEC3PARAM existe

NSEC3

Gestion des réponses négatives, mais lutte aussi contre les "énumérations de zone". NSEC3 utilise une fonction de hachage qu'on utilise sur les noms renvoyés. on envoie le hash du prochain nom de la zone => On ne peut pas reconstruire la zone entière.

NSEC3PARAM => Utilisé par un serveur DNS pour calculer et déterminer les records NSEC3 à ajouter dans les réponses négatives.

Le record contient:

Algorithme de hachage

Flag "opt-out" = délégations signés ou pas

Itérations = combien de fois l'algo doit être appliqué

Salt = valeur de Salt pour le calcul du hash

EXEMPLE:

```

0 NSEC3PARAM 1 0 100 85A9F28CC4264523

```

1 = SHA-1

0 = Opt-out

100 = itérations

85A9F28CC4264523 = Salt

Pour le record NSEC3:

```

C0O2TRHFHTFE2HOV1R6P1GGOJ58J5V56.esit.be. 3600 IN NSEC3 1 0 100 85A9F28CC4264523 Q3D1L6GDFG9KAVMVPQANF1N9VI6G1C60 A NS SOA AAAA RRSIG DNSKEY NSEC3PARAM

```

Rouge = données reprises dans le record NSEC3PARAM

Les noms de domaine sont hashés -> quand le resolver voudra vérifier la réponse négative -> génère le hash du nom demandé, c-à-d h2 (on doit donc vérifier que h2 est entre h1 et h3) h1 et h3 = zones en bleu. Il est conseillé de remplacer régulièrement le Salt.

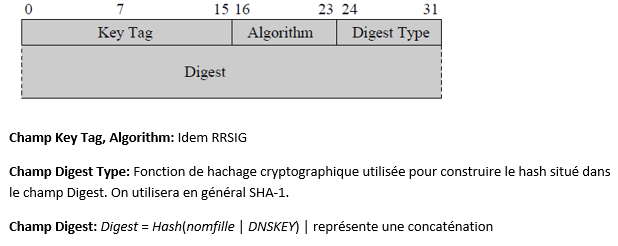
Enregistrement DS (Delegation Signer)

= Chaine de confiance dans les liaisons entre serveurs DNS parents et enfants. DS placé dans la zone parent et lié au record DNSKEY de la zone enfant. Contient un hash de la clé KSK (zone enfant), stocké avec le record NS qui indique la délégation vers la zone enfant. Enregistrement DS est signé avec le reste des données de la zone parente.

Ce record est demandé par le resolver au serveur autoritaire dès qu’il doit vérifier une signature d’une zone enfant. De cette façon, la zone parent indique qu’il peut faire confiance au serveur en renvoyant un hash de la clé par la zone parent. Le resolver peut donc faire confiance à ce serveur.

On retrouve le schéma à chaque maillon de la chaîne : un record DS qui permet d’avoir confiance en une KSK, elle permet d’avoir confiance en l’ensemble des clés de la zone (KSK et ZSK). ZSK permettent d’avoir confiance dans les autres records.

Record DS =



EXEMPLE :



42235 : key ID ( ID de la DNSKEY qui correspond à la KSK de la zone enfant)

8 : numéro de l’algorithme utilisé pour DNSKEY (RSA/SHA-256)

1 : SHA-1

2 : SHA-256

ZSK

Clé publique connue des serv récursifs pour vérifier les hash. RRSIG ont une période de validité (pas les clefs) mais pour une question de sécurité il vaut mieux renouveler la clé tous les mois.

(serv DNSSEC a un mécanisme de rotation de clef chaque mois).

* 1ère technique : génère une clé et signe la zone avec la clé en laissant l’ancienne dans le fichier (va être supprimée avec le temps d’expiration)
* Pré publier une nouvelle clé. Deux clés : clé active et passive. Une seule des deux est utilisé pour signer les records, l’autre pas utilisée. On ajoute la nouvelle clé à la fin du TTL de l’ancienne. Au pire on peut mettre les deux clés dès le début du mois. En fin de mois, la zone est signée avec la clé passive (clé passive signée par KSK)

KSK

= Clé privée utilisée pour signer une zone ZSK. Seulement utile pour un record DNSKEY. Est un maillon de confiance. Signatures des clés ZSK ont une durée de validité d’1 an. Remplacement plus complexe car on doit aussi agir dans la zone parent.

1. On ne sait pas quand le parent changera ou publiera le nouveau record DS.
2. On doit maintenir l’ancienne KSK le temps que le TTL du record DS soit expirée => problèmes de cache.

Pour les parents avec **UN SEUL** record DS :

1. Mettre une nouvelle clé KSK dans la zone enfant
2. Signer les ZSK avec la KSK avec les anciennes signatures
3. Attendre que les caches aient la nouvelle version de la zone avec les nouvelles signatures
4. Publier le record DS chez le parent, remplace l’ancien.
5. Attendre que le parent publie le DS
6. Attendre le plus longtemps entre le TTL de l’ancien et du nouveau
7. Retirer l’ancienne KSK et supprime les anciennes signatures des ZSK

Pour les parents avec **PLUSIEURS** records DS :

1. Nouvelle KSK dans la zone enfant
2. Signer les ZSK avec la nouvelle KSK en gardant les anciennes signatures
3. Publier le nouveau record DS en gardant l’ancien DS
4. Attendre que le parent publie le nouveau record DS
5. Attendre le plus longtemps entre le TTL de l’ancien et le nouveau
6. Retirer l’ancienne KSK et supprimer les anciennes signatures des ZSK
7. Supprimer l’ancien record DS du parent

Calcul de la signature = Kzsk(Hash(RRSIGDATA | RR(1) | RR(2) |…))

RRSIGDATA = concaténation de tous les champs du record RRSIG sauf le champ signature

RR(x) = nomdetenteur | type | classe | TTL | TailleRDATA | RDATA

Nomdetenteur = nom du signataire en canonique, correspond au champ Signer’s Name du RRSIG

Classe = classe du RRSIG

Type fait partie des types listés dans Type Covered Field

TTL = TTL du RRSIG

**Trust Anchor**

Pour que la chaine de confiance puisse être appliquée, nous devons introduire le **Trust Anchor**.

* Pour faire confiance on fils, on demande au père
* Le Trust Anchor est la première clé de la chaine à qui on fait confiance. La clé est la clé publique associée à la KSK de la racine. Tous les serveurs sont censés connaître la clé entrée manuellement.
* Élément critique de la chaine de confiance, car si un hacker trouve la clé privée, toute la chaine est compromise.

**Mise en place d’un îlot sécurisé**

Il y’a encore des serveurs qui n’ont pas déployé DNSSEC. Il y’a donc des îlots sécurisés qui ne peuvent pas faire valider leurs clés par leurs serveurs autoritaires, mais qui développent quand même des échanges sécurisées avec les sous-domaines.

Pour le mettre en place, on doit introduire la clé publique associée à la KSK de la zone mère dans les différents serveurs. Pour indiquer si un serveur fille est sécurisée, on doit introduire le record DS de la fille dans le fichier de zone autorite de façon manuelle.

**DNSSEC Lookaside Validation (DLV)**

= Alternative à la gestion manuelle des trust anchors. Idée = Dissocier la racine du DNS et la racine de signature DNSSEC. Avec DLV, plusieurs racines peuvent signer le même domaine (racine DLV signe .org et une autre qui signe site.org). On peut utiliser n’importe quel algo avec le resolver DLV. Beaucoup de détracteurs car elle retarde la signature de la racine. Quand la racine est signée, la clé de racine suffit en tant que trust anchor. => C’est pour ça le fait qu’il y’a peu de TLD qui ont leur trust anchor publiée dans le dépôt DLV de l’ISC (Internet Systems Consortium).

**IMPORTANT**

DNSSEC peut causer des pbs de compatibilité avec l’hardware réseau qui gèrent le DNS. Les pare-feu, routeurs et autres doivent être compatible avec DNSSEC. PB = paquets DNSSEC sont plus volumineux que le DNS (limité octets). En raison des limites de taille de MTU, UDP ne peut pas toujours prendre en charge les paquets DNSSEC. => TCP ? **NON**, ça alourdit le trafic. Certains périphériques ne prennent pas DNS en charge sur TCP => longue phase de test niveau matériel pour le bon fonctionnement du DNSSEC.

**PASSAGE DNSSEC -> BIND9**

1. Activer DNSSEC sur le serv BIND9 => modifier fichier de config



Quand le DNSSEC est activé, on peut signer les fichiers de zone. => Utilise l’utilitaire zonesigner. Si pas présent -> on installe dnssec-tools.

1. Signer le fichier de zone

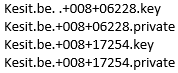


-genkeys = génération de clé (KSK et 2 clés ZSK) + signer la zone esit.be qui correspond à esit.be.zone.

-usensec3 = utilisation de record NSEC3 au lieu de NSEC

=> On obtient :

Deux clés ZSK (1 privée 1 publique pour chaque) :



Une clé KSK (1 privée 1 publique) :



Record DS pour la zone parent :



Fichier qui réf les clés de la zone esit.be :



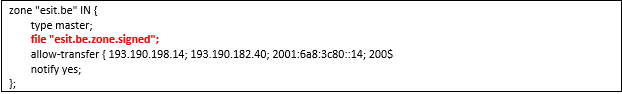
Fichier de zone signé :



1. Dès que la zone est signée, on doit la charger en modifiant la config de la zone (/etc/bind/named.conf.local)



On remplace avec le nouveau fichier : esit.be.zone.signed



**Le fichier de zone non signé doit être conservé => On y apportera les modif à la zone => à chaque fois resignée.**

1. Changer le record DS dans la zone parent. Procédure à suivre dépend du provider

Plusieurs points importants de la gestion DNSSEC

1. Rotation automatique des ZSK chaque mois et remplacement manuel KSK chaque année. Si on passe au-delà de la validité -> zone plus accessible (demande DNS return SERVFAIL)

Outil fournit avec dnssec-tools -> roolerd. Fichier du deamon peut être réalisé avec l’utilitaire rollinit **OU ALORS** être modifié dans /etc/dnssec-tools/dnssec-tools.rollrec -> EXEMPLE :



1. Modif du fichier de zone. **TOUTES LES MODIFS SONT FAITES DANS LE FICHIER NON SIGNÉ.**

EXEMPLE :



**IPv6**

**Intro**

Adresses en 32 bits (de 0 à 255 décimales => exemple : 192.168.103.14)

* Permet 2^32 adresses (4 294 967 296) -> 4 milliards. Hôtes qui ont besoin d’une IP double chaque année -> nombre d’adresses dispo diminue beaucoup.

En 2011, 5 derniers blocs d’IP distribués par IANA, les IP restantes dispo sont gérés par les RIR. Maintient de l’IPv4 en libérant les adresses non utilisées des : adresses en format CCIDR, affection d’adresses réservées, le NAT et la demande de clients ayant reçu des IP en /8.

* Quand même obligé de trouver un nouveau moyen pour se libérer des limites imposés par IPv4 => routage plus efficace.

Architecture IPv6 définie dans RDC 2373 (IPv6 Addresing Architecture) = Adresses en 128 bits (16 octets). Pas facile de représenter une IPv6

**Représentation de l’IPv6**

* Pour éviter les décimaux énormes, IPv6 est en hexadécimal avec des « : » entre les blocs. 8 blocs de 2 octets :



* Chaque bloc est séparé des autres, les 0 non significatifs peuvent être éliminés :



* Si plusieurs blocs consécutifs sont composés de 0, ils peuvent être abrégés :

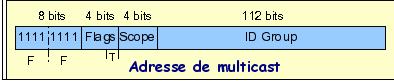




* **Utilisation du « :: » qu’une seule fois dans une adresse IPv6**

**IPv6 et les types d’adresses**

* Unicast : Identification d’une seule interface. Si un paquet est envoyé à une adresse Unicast, elle est reçue par l’interface dépositaire de l’adresse.
* Multicast : Groupe de plusieurs interfaces de plusieurs nœuds. Paquet envoyé vers une adresse multicast est reçue par tous les membres du groupe. Adresse multicaast comment par : « ff00 ::/8 »



Bit T = permanence de l’adresse, T=0 (adresse permanente), T=1 (adresse temporaire)

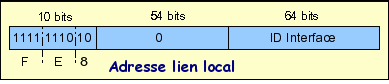
Scope = portée de diffusion , 0(réservé), 1(nœud), 2(lien), 3(sous-réseau),5(site),8(organisation),E(global),F(réservé)

* Certaines adresses sont prédéfinies  -> considérées comme permanentes :
  + Ff02 ::1 -> toutes les machines du lien
  + Ff02 ::2 -> tous les routeurs du lien
  + Ff05 ::2 -> tous les routeurs du site
  + Ff02 ::1 :ffxx :xxx -> tous les noeuds se terminent par xx :xxx dans leur IP
* Anycast : Identifie un groupe d’interfaces de plusieurs nœuds (comme Multicast). Quand un paquet est envoyée à une adresse Anycast -> reçue par un des membres du groupe anycast (souvent le plus proche en terme de routage).

En IPv6, type d’adresse broadcast n’existe plus.

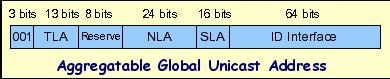
**ADRESSES PARTICULIÈRES EN IPv6 :**

* + Adresse de bouclage :
    - ::1
  + Adresse inconnue composée de 0
    - ::
  + Adresse de lien local
    - Donnée aux cartes d’interface dynamiquement. Valide uniquement dans un espace local qui ne traversent pas de routeur. Préfixe = « fe80 ::/10 » Structure :



Sous windows, ipconfig apparaît les adresses de link local, une par interface. IP indépendante de toute config. Sert à adhérer aux divers groupes multicasts sur le lien, remplacent le broadcast IPv4 et ARP.

* + Adresse de site local :
    - Adresses privées IPv4. Routables mais ne peut pas être utilisé sur Internet. Préfixe = fec0 ::/10. Après les 10 premiers bits, les 54 suivants définissent l’adresse de network. La RFC qui a introduit est passé en deprecated -> chaque équipement aura son adresse publique -> adresse IPv6 locales -> POUBELLE
    - En 2005, une RFC introduit les adresses de type Unique Local Address. Préfixe = fc00 ::/7. 8ème bit L est assigné à la valeur 1. Ensuite ID global sur 40 bits, ID de sous-réseau sur 16 bits + interface sur 64 bits. => Possibilité de pouvoir utiliser un ID global qui permet l’interconnexion de sites sans changer l’ID des réseaux.
  + Adresse utilisable sur Internet
    - Structure :

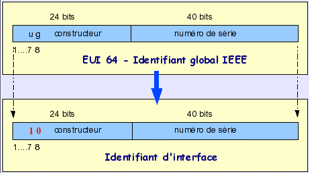


* + - Préfixe sur 3 bits, valeur correspond à son adresse unicast publique.
    - Top Level Agegator -> différents blocs libérés par IANA -> premier porte sur numéro 1. 16 bits correspondants seront donc : 0010000000000001 et 2001 en hexa. 32 bits suivants dont 8 de poids forts se trouvent répartis comme :
      * 13 bits comme SUB-TLA, 19 bits comme NLA.
    - **CETTE RFC EST MAINTENANT ARCHIVÉE**
    - IPv6 distribuées dans le premier bloc 2001 :: aux RIR sont des blocs /23 -> RIR distribueront les IP aux LIR en /32 => Belnet reçoit un /32.
* Belnet dispose de 16 bits pour répartir les adresses aux users. = > Adresses /48 distribuées aux gros users.
  + - * EXEMPLE : L’HELHa Tournai a reçu 2001:06a8:3480::/48. Sur une adresse de 128 bits, le site final gère 80 bits dont 16 pour identifier un réseau et 64 bits restant l’host sur le réseau. La HELHa pourra donc connecter 2^16 réseaux et chaque réseau 2^64 hosts. Les IP sont attribués via un serveur DHCP.

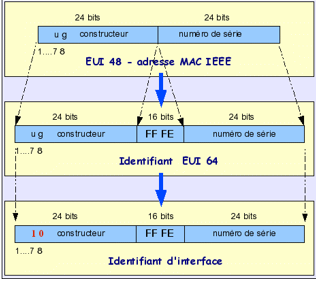
**Création des adresses de lien**

Dans ce chapitre, on analyse la façon dont les 64 bits de l’ID interface seront choisis (adresses de lien local sont attribués de façon automatique au démarrage de toute interface).

1ère solution = se baser sur une adresse physique dans un format EUI-64 de l’IEEE. On insère donc l’adresse physique dans les 64 bits de l’IPv6.



2ème solution = basé sur les adresses physiques sur 48 bits comme les adresses MAC des cartes réseaux => Format EUI-48 de l’IEEE. Pour combler les 64 bits, on fait cette opération :



**Les adresses anonymes**

Adresses fabriquées à-p-d l’EUI-64 n’était plus anonyme. Sous XP, (avec un routeur qui annonce un réseau), l’adresses normale est auto configurée mais une autre adresse l’est aussi. Cette adresse est dite « anonyme » dont la partie hôte est choisie au hasard (remplace EUI-64). Fonction désactivable.

Chaque préfixe, chaque interface reçoit une IPv6 supplémentaire :

<préfixe><EUI-64>

Sous XP/2003, on ajoute une IPv6 anonyme en plus :

<préfixe><64 bits au hasard>

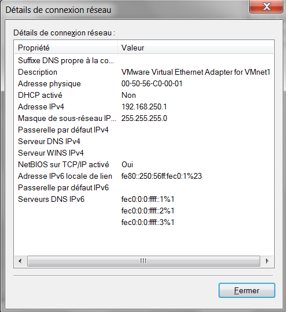
Pour les connexions sortantes, on utilise l’IP anonyme.

Pour les connexions entrantes, on peut joindre sur les 2 IP.

Quel que soit le réseau de la machine, l’EUI-64 reste toujours le même => Quelque soit le réseau et la période de connexion, on peut identifier la machine elle-même.

L’adresse anonyme de Microsoft est censé palier à ce « problème », l’anonymat en matière de réseau est largement théorique.

EXEMPLE :



Adresse MAC de la carte réseau : 00-50-56-C0-00-01

Adresse de lien : 00-50-56 FFFE C0-00-01

On a donc 0250:56FF:FEC0:0001 (2 pour bit ‘u’ positionné à 1)

Adresse de lien local sera donc : fe80:250:56ff:fec0:1%23 (%23 représente l’interface associée a l’adresse)

**Solicited Node Address, Neighbor Discovery, Router Advertissement**

Pour chacune des IPv6, chaque interface adhère à un groupe multicast calculée en fonction de l’interface. Adresse = solicited node address => adresse multicast de lien local se présente comme ceci : **FF02:0:0:0:0:1:FFXX:XXXX**



Les X dans l’adresse représente la partie variable de l’adresse. Contient les 24 derniers bits de l’IP d’origine. (3 octets ou 6 nibbles de droite). Donc, si l’adresse à-p-d EUI-64 est utilisé, toutes les IPv6 de l’interface se lient donc à une seule solicited node address (car EUI-64 ne change pas même si la partie réseau change).

En connaissant l’IPv6, on peut directement reconnaître la solicited node address. Pour découvrir l’adresse MAC relative à une IPv6 connue, il faut envoyer un message ND (Neighbor Discovery) vers cette adresse en utilisant l’adresse de lien local.

**Intérêt de l’utilisation de la solicited node address :**

La méthode fait appel à plusieurs groupes multicast auxquelles peu de nodes adhèrent en même temps. Multicast sur un segment Ethernet de base = broadcast niveau 2. **MAIS** niveau équipement niveau 2 et 3 qui gère le multicast (chaque adhésion multicast IPv6 crée une adhésion niveau 3, par ND, mais aussi au niveau 2 par broadcast Ethernet), utilisation du ND et donc du multicast permet de réduire **ÉNORMÉMENT** le nombre de broadcasts sur le réseau et de limiter le trafic de résolution des adresses MAC, convertit le broadcast en quasi-unicast.

Au démarrage, une interface a une adresse de lien local (avec EUI-64) et se lie à la solicited node address correspondante. Si il n’y a pas de routeur sur le lien, il n’y a pas d’adresse attribuée automatiquement. Le node cherche la présence d’un routeur avec un multicast approprié : Le Router Solicitation.

**Si un routeur est présent et configuré**, il répondra au message par un RA (router advertisement). Puisque le message a été expressément demandé, il s’agira d’un Solicited Router Advertisement. Il contient les préfixes disponisbles sur le réseau au format CIDR ainsi que sa propre adresse. Avec l’adresse du réseau, la machine peut accoler le réseau et son EUI-64 pour former une IPv6 routable et unique. Le routage est possible car la machine utilisera l’adresse du routeur fournie dans le RA. Si il y’a plusieurs routeurs, plusieurs IP seront fabriquées et plusieurs routes par défaut seront rajoutées.

**L’adresse est fabriquée comme ceci :** <64 bits du réseau, annoncé par le routeur><EUI-64 : 64 bits hosts, définis à-p-d de la MAC>

Tout routeur correctement configuré émet régulièrement sur le réseau des RA pour annoncer les informations. Puisque personne ne les réclame, il s’agit d’Unsolicited Router Advertisement.

Il est possible de configurer des IP statiques et indépendantes de l’EUI-64, on peut aussi désactiver l’auto-config des adresses et de la route par défaut. **PRATIQUEMENT INDISPENSABLE POUR LES ROUTEURS.**

Quand un host détecte un routeur, une route par défaut est automatiquement ajoutée. Cette adresse est une adresse de lien (pas une adresse globale). Choix justifié car une adresse de lien ne change pas. Une adresse globale peut être modifiée par l’administrateur du réseau.

Exemple de la création de la SNA :

2 hosts sur un segment Ethernet, A et B. Host A a une link local fe80:100:100:100::A/64 . Tandis que Host B a une link local fe80:100:100:100::B/64. A veut rentrer en contact avec B.

1. A doit connaitre la MAC de B. En IPv4, la récupération est faite avec le protocole ARP (broadcast). Host A peut envoyer un paquet vers l’adresse de multicast à laquelle Host B est joint. Il suffit pour l’Host A de prendre le préfixe **ff02:0:0:0:0:1:ff00::/104** et d’ajouter les 24 bits de poids faible de l’adresse de lien d’Host B = **ff02:0:0:0:0:1:ff00:000B.**
2. Host B est déjà configuré au niveau de son solicited node address et a rejoint le groupe **ff02:0:0:0:0:1:ff00:000B** en utilisant MLD (substitut IPv6 d’IGMP). Host B, comme membre du groupe, recevra un NS (Neighbor Solicitation) de Host A.
3. Host B répond avec sa MAC dans la partie payload du Neighbor Advertisement Message.

**Attribution des IP globales**

Le mécanisme d’auto configuration des IP en mode stateless prend aussi en charge les adresses globales grâce à des messages de type Router Solicitation et Router Advertisement avec les routeur voisins.

En stateless, on peut configurer les IP de façon automatique au travers de la découverte des préfixes. Pour chaque préfixe renseigné sur le routeur pour une interface donnée, on retrouve deux flags :

* + - Flag Autonomous : Instruit l’host qui reçoit la donnée en générant une adresse automatique à-p-d du préfixe. Si le flag = false, pas d’adresse générée sur base du préfixe.
    - Flag OnLink : Instruit l’host qui reçoit la donnée que le préfixe est OnLink et qu’il doit être utilisé dans la config automatique, e.g. : dans la table de routage de l’host.

Dans cette solution, aucune config n’est envisagée pour les mécanismes de résolution de nom. On doit donc soit configurer les informations manuellement soit utiliser un serveur DHCPv6 qui se limite à attribuer des IPv6 stateless et d’attribuer des options comme : serveur DNS, serveur de temps, etc…

Dans la config des messages Router Advetisement sur un routeur, on peut retrouver deux flags Managed Configuration et Other Parameters Configuration -> permet de définir la façon dont les IP et paramètres sont obtenus automatiquement par un serveur DHCPv6. Les deux flags sont **M flag** et **O flag**.

Effet des flag envoyées d’un routeur vers un host :

* Flag M et O à 0 : Correspond à un réseau sans serveur DHCPv6. Paramètres devront être définies manuellement.
* Flag M et O à 1 : DHCPv6 utilisé pour fournir les paramètres et l’IPv6 -> DHCPv6 statefull.
* Flag M 0 et O 1 : DHCPv6 utilisé pour assigner les paramètres, IPv6 attribuée automatiquement par les mécanismes de découverte des routeurs voisins -> DHCPv6 stateless
* Flag M 1 et O 0 : Adresse attribuée par DHCPv6, paramètres non définies.
* On peut donc se retrouver avec deux IPv6 pour le même préfixe. Si on définit sur le routeur un préfixe avec un flag autonomous avec un flag M à 1, deux adresses sont générées. L’une fournit par le DHCPv6 et l’autre pour le préfixe fourni.

**Tables de routages sur les hosts**

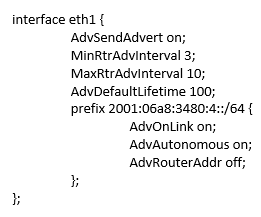
En DHCPv4, il est possible de définir des routes et une passerelle par défaut comme options à fournir par le client. Dans les premières normes DHCPv6, les options ne sont pas présentes. Les options ont étés ajoutées mais pas implémentées dans tous les serveurs et/ou clients DHCP.

Passerelle par défaut est ajoutée automatiquement lors de la découverte des routeurs. Pour les tables de routage, elles sont générées sur base des préfixes définis sur les routeurs. Il suffit de mettre le flag autonomous à 0 et le flag OnLink à 1 pour chacun des préfixes.

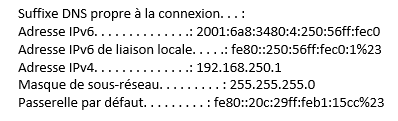
S’il y’a plusieurs routeurs sur le même segment, on peut définir un ordre de préférence pour les différentes routes fournies si des routes peuvent être redondantes. L’Host peut donc voir un routeur comme routeur primaire ou routeur secondaire. On donne donc trois valeurs : High, Medium et Low.

Pour la passerelle par défaut, il est possible, s’il y’a plusieurs routeurs, de supprimer la passerelle de l’host pour un routeur déterminé. Il suffit de placer le default lifetime à 0 pour le routeur concerné. Un host qui recevra la propriété écartera l’adresse de lien du routeur si le lifetime de celui-ci = 0.

Pour aider dans les test de config auto (cas où un routeur est présent sur le lien local), on configurera un Debian avec le package radvd. Voilà la config du routeur (/etc/radvd.conf)



Quand le service sera démarré, on obtiendra pour cette interface, les informations suivantes :

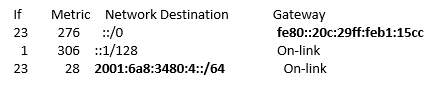


L’adresse de liaison locale existe toujours. On retrouve, en plus des autres, une adresse générée automatiquement sur base du préfixe renseigné par le routeur. La génération de l’adresse est possible grâce au flag ‘AdvAutonomous’ mis à 1. On rajoute au préfixe l’EUI-64 (identique à celui qui est utilisé pour l’adresse de lien)

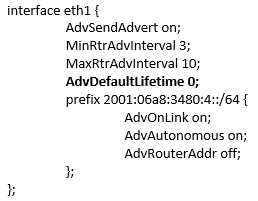
Une adresse de passerelle est ajoutée = adresse de liaison locale du routeur

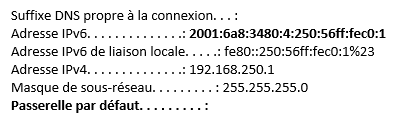


Si on analyse la table de routage, on peut voir :



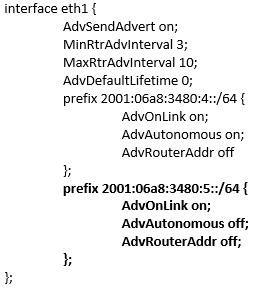
Route par défaut qui correspond à l’adresse de liaison du routeur. On retrouve aussi la route associée au préfixe fourni par le routeur. S’il y’a plusieurs routeurs présents et qu’on souhaite supprimer une route par défaut pour un routeur, on peut supprimer cette route en définissant le temps de vie de l’information fournie par le routeur à 0.





L’adresse de scope global existe toujours mais la passerelle par défaut est maintenant absente. Si l’on veut ajouté des routes supplémentaires associées à d’autres préfixes (sans générer d’adresses automatique basé sur ce dernier), on peut jouer sur le flag AdvAutonomous que l’on peut placer à 0

Exemple :



Si on regarde les adresses automatiques, on remarque qu’aucune adresse automatique n’a été générée pour 2001:06a8:3480:5::/64 mais on peut retrouver cette adresse dans les tables de routage.

