Université de Limoges
Faculté des Sciences et Techniques
Tuteur universitaire : Julien Iguchi-Cartigny

Domain Name System : Attaques et sécurisation

Hamza HMAMA

Guillaume LUCAS

Limoges, le 16 août 2012

Licence

Remerciements

Nous adressons de sincères remerciements à Julien Iguchi-Cartigny pour nous avoir donné l'opportunité d'étudier le présent sujet.

Nous remercions vivement Florian Maury pour ses relectures et pour les nouvelles pistes de réflexion qu'il a fait germer.

Sommaire

1 Introduction

Un projet, d'une durée approximative de trois mois, sur un thème volontairement choisi par les étudiants est à réaliser lors du deuxième semestre de la troisième année de la licence informatique à la faculté des sciences et des techniques de Limoges. Ce projet donne lieu à l'écriture d'un rapport et à une soutenance.

Pour ce projet, nous avons choisi de travailler sur le système de noms de domaine (Domain Name System abrégé DNS). Il s'agit, pour nous, d'avoir une meilleure compréhension du système, de ses mécanismes ainsi que de ses faiblesses.

Notre intérêt pour DNS vient du fait qu'il est un système vital de l'internet moderne. En effet, toute transaction, qu'il s'agisse, entre autres, d'une recherche sur un moteur de recherche, de l'envoi d'un courriel ou bien encore d'une visioconférence, toutes commencent par une requête DNS.

Afin de bien comprendre les rouages du système, nous avons décidé, dans un premier temps, de mettre en place une maquette d'un système DNS complet et de tester, sur cette maquette, les principales faiblesses connues du système DNS.

Dans un deuxième temps, nous avons élaboré une deuxième maquette qui met en œuvre le DNS sécurisé. Nous avons réalisé, sur cette maquette, un ensemble de tests de sécurité visant à démontrer les apports et les nouvelles faiblesses induites par DNSSEC.

Ce rapport se propose de synthétiser notre projet. Dans un premier temps, il abordera DNS à travers une série de présentations d'une complexité croissante et à travers une mise en œuvre pratique. Dans un deuxième temps, ce rapport abordera DNSSEC, à nouveau à travers des présentations et une mise en œuvre pratique.

2 DNS

2.1 Présentation succincte

Un nom de domaine est un identifiant unique pour un domaine internet, c'est-à-dire pour un ensemble de machines reliées à internet mais partageant des caractéristiques communes [1]. Par exemple : unilim.fr. est le nom de domaine commun à toutes les machines dépendantes plus ou moins de l'université de Limoges [2].

Les noms de domaine peuvent être enregistrés par des particuliers, des entreprises, des organisations ou des administrations publiques. Les noms de domaine n'appartiennent jamais à une entité mais sont loués pour une période bien définie [3].

Le DNS (Domain Name System), traduit en « système de noms de domaine », conçu dès 1982, est la réunion de deux services liés aux noms de domaine :

- L'enregistrement d'un nom de domaine : assure l'unicité des noms et l'association entre un nom et son propriétaire légal.
- La résolution d'un nom de domaine : permet à une machine d'obtenir des informations (adresse IPv4, adresse IPv6, relais de courrier, ...) en échange d'un nom de domaine. Cette résolution se fait avec un protocole réseau nommé, lui aussi, DNS.

Ces deux services sont hiérarchiques (et non pas centralisés comme on l'entend souvent !), redondants et distribués. Hiérarchiques, en effet, car la totalité de la chaîne ne dépend pas d'un unique acteur. Par exemple : le nom de domaine unilim.fr n'est pas sous le même contrôle administratif que google.fr [4]. Voir la figure 1. Distribués et redondants car la totalité des informations n'est pas stockée sur un même serveur ou par une même entité.

1. Pour les lecteurs désireux d'aller plus loin, sachez qu'un nom de domaine n'est unique que dans une racine/hiérarchie donnée (comparable à un namespace en programmation). Dans tout cet exposé, nous parlons, de manière implicite, de la racine la plus couramment utilisée, celle de l'ICANN mais d'autres racines existent comme OpenNIC ou celles mises en place dans des organismes privés.

2. Même si cette notion n'est plus aussi forte qu'au commencement (il est possible d'avoir un .net sans être opérateur de réseau, par exemple), elle reste valable.

3. Pour les lecteurs désireux d'aller plus loin, sachez que cette affirmation est vraie dans la racine ICANN mais qu'elle peut varier dans une autre racine car il ne s'agit pas d'une question technique mais d'un choix.

4. Pour les lecteurs avancés, il faut comprendre que bien que ces deux noms dépendent de l'AFNIC et du droit français, ce contrôle est minimal (pas de droit de regard sur le contenu de la zone, par exemple). De plus, nous cherchions à illustrer le principe que Google n'a aucun droit de regard sur le contenu de la zone de l'université de Limoges.

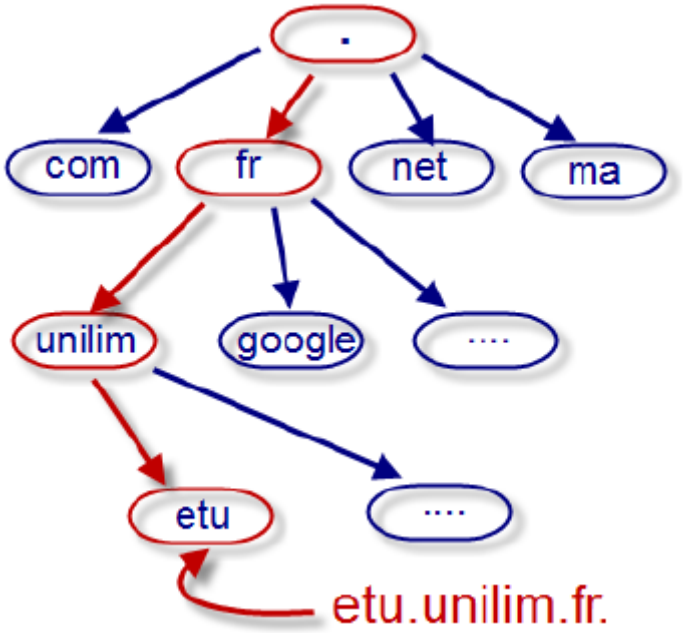

FIGURE 1 – La hiérarchie des noms dans le DNS.

L'analogie la plus commune est de voir le DNS comme un annuaire téléphonique : ce dernier permet de traduire un nom humainement compréhensible et facilement mémorisable par un numéro de téléphone qui est spécifique à la topologie du réseau téléphonique et plus difficilement mémorisable.

2.2 Présentation détaillée

Avant l'utilisation de DNS, à partir de 1985, les associations entre les noms de domaine et les adresses des machines étaient réalisées manuellement dans un fichier « HOSTS.TXT » qui était distribué par le SRI-NIC (Stanford Research Institute - Network Information Center).

Ce modèle fonctionnait car le réseau était un réseau restreint dédié aux militaires, aux universitaires et à la recherche. Le système d'adressage (pré-IP) sur 1 octet permettait également la constitution d'un tel listing.

Puis le nombre d'hôtes connectés augmenta, amenant un coût de distribution élevé du fichier HOSTS.TXT ainsi qu'une importante proportion de fichiers non mis à jour. Ce constat mit un terme au modèle HOSTS.TXT et réclama la naissance d'un nouveau modèle de résolution de noms décentralisé. Plusieurs projets tentèrent de remplacer HOSTS.TXT comme IEN 116 [**?**] ou Grapevine. Le premier fût jugé comme étant trop simple alors que le second fut jugé comme étant trop compliqué. DNS a donc pris le dessus.

Concernant les dates du projet DNS, les pères de ce projet ont retracé, en 1988, dans leur papier « Development of the Domain Name System » [5], les étapes suivantes :

- 1982 : Constat de l'inefficacité de HOSTS.TXT.
- 1983 : Normalisation de la première version de DNS dans les RFC 882 et 883.
- 1984 : Première implémentation de DNS : BIND.
- 1987 : Normalisation du DNS tel que nous le connaissons aujourd'hui dans les RFC 1034 et 1035.

On peut avoir deux écritures pour un même nom de domaine : l'écriture absolue ou l'écriture relative. Un nom de domaine absolu (ou nom de domaine totalement qualifié, de l'anglais Fully Qualified Domain Name, FQDN) est un nom de domaine complet qui comprend donc tous les labels, y compris celui de la racine, « . ». Exemple : unilim.fr n'est pas un FQDN alors que unilim.fr. en est un. Un domaine relatif ne se termine donc pas par un point et est interprété comme étant relatif au domaine dans lequel se situe la machine. Exemple : si je suis sur une machine du domaine unilim, le nom www est équivalent au FQDN www.unilim.fr.. Si je suis dans un autre domaine, je ne suis pas sûr de sa signification [6]. Dans la suite de ce rapport, tous les noms seront des FQDN même s'il peut arriver que le point final soit parfois manquant.

5. Une numérisation de la version papier originale est disponible à cette adresse : `http://cseweb.ucsd.edu/classes/wi01/cse222/papers/mockapetris-dns-sigcomm88.pdf`. Une présentation dudit papier est également disponible à cette adresse : `http://www.bortzmeyer.org/bind-dns-history.html`.

6. Il y a donc matière pour s'amuser avec les clauses « search » (contenues, entre autres, dans le fichier resolv.conf) et les algorithmes associés.

2.2.1 Enregistrement d'un nom de domaine

Au sommet de la hiérarchie, on trouve l'Internet Assigned Numbers Authority (IANA), une entité états-unienne qui est une unité opérationnelle de l'ICANN (Assigned Names and Numbers) depuis le 1er janvier 1999. L'ICANN est une entité états-unienne sous contrat avec le département du commerce américain.

L'ICANN délègue la gestion de chaque domaine de premier niveau (à l'exception de int., arpa. et les autres zones critiques comme root-servers.net. [7]) à des entités distinctes [8] qui sont appelées registries (pluriel de registry, registres ou opérateurs de registre en français). Ils ont pour responsabilités de :

- Maintenir la base de données de tous les sous-domaine du domaine de premier niveau dont ils ont obtenu la responsabilité.
- Générer, à partir de cette base, les fichiers de zones correspondant aux domaines de premier niveau dont ils ont le contrôle.
- Définir et contrôler l'application des règles d'attribution des sous-domaine.

Les registries (AFNIC, Afilias, ...), souscrivent des contrats avec autres organisations (commerciales la plupart du temps) appelées registrar (bureau d'enregistrement en français). Ces organisations (Gandi, OVH, ...) sont accréditées par les registries pour louer des noms de domaine. Les noms de domaine sont donc enregistrés auprès de ces organismes. Lors de chaque nouvel enregistrement ou renouvellement, les registars demandent la mise à jour de la base de données du registre concerné grâce au protocole Extensible Provisioning Protocol [9].

7. Source : Point 10 - Operations - What ICANN Does - Staff Draft - Version 1.1 - 10 March 2002 - `http://archive.icann.org/en/general/toward-mission-statement-07mar02.htm`.

8. Même si certaines entités concentrent beaucoup de pouvoir. Exemple : Verisign est l'opérateur technique de la racine pour le compte de l'ICANN ainsi que des TLD net. et com. De plus, Verisign a d'autres rôles dans d'autres TLD comme edu ou gov.

9. Pour plus d'informations, voir : `https://en.wikipedia.org/wiki/Extensible_Provisioning_Protocol`.

2.2.2 Résolution d'un nom de domaine

Un nom de domaine est découpé en labels. Chaque label ne correspond pas obligatoirement à une zone. Par exemple : dans le nom pop.etu.unilim.fr., nous avons cinq labels (« . », « fr. », « unilim.fr. », « etu.unilim.fr. » et « pop.etu.unilim.fr. ») mais seulement trois zones : « . », « fr. », « unilim.fr. ».

Chaque zone est distribuée sur le réseau par au moins un serveur faisant autorité. Un serveur faisant autorité est un serveur qui comprend le protocole DNS et qui répond uniquement aux requêtes concernant la zone pour laquelle il possède des informations ou pour une sous-zone (dans ce cas, le serveur ne connaît toutefois que les informations de zone-cut (enregistrement(s) NS) et les glue-records (enregistrement(s) A/AAAA correspondant au(x) NS).

Exemple : si un client demande à l'un des serveurs faisant autorité pour fr., l'enregistrement « A www.unilim.fr. ? », celui-ci répondra « unilim.fr. NS limdns.unilim.fr. » qui signifie "je ne gère pas la zone unilim.fr. Mais je sais quel serveur fait autorité sur cette zone, va te renseigner auprès de lui".

Ce type de serveur a donc un comportement itératif : il aide le client uniquement pour une partie de la résolution du nom (le serveur faisant autorité sur fr. ne donnera pas directement une réponse pour etu.unilim.fr, par exemple).

Un client qui souhaite résoudre un nom de domaine s'adresse à un serveur dit résolveur ou serveur récursif. Celui-ci tente de trouver une réponse à la requête du client. Pour cela, il interroge le serveur faisant autorité sur les zones demandées en partant de la racine. Voir la figure 2 pour une explication en image.

Ce serveur a un comportement récursif : il prend en charge la totalité du processus de résolution d'un nom et transfert la réponse finale au client.

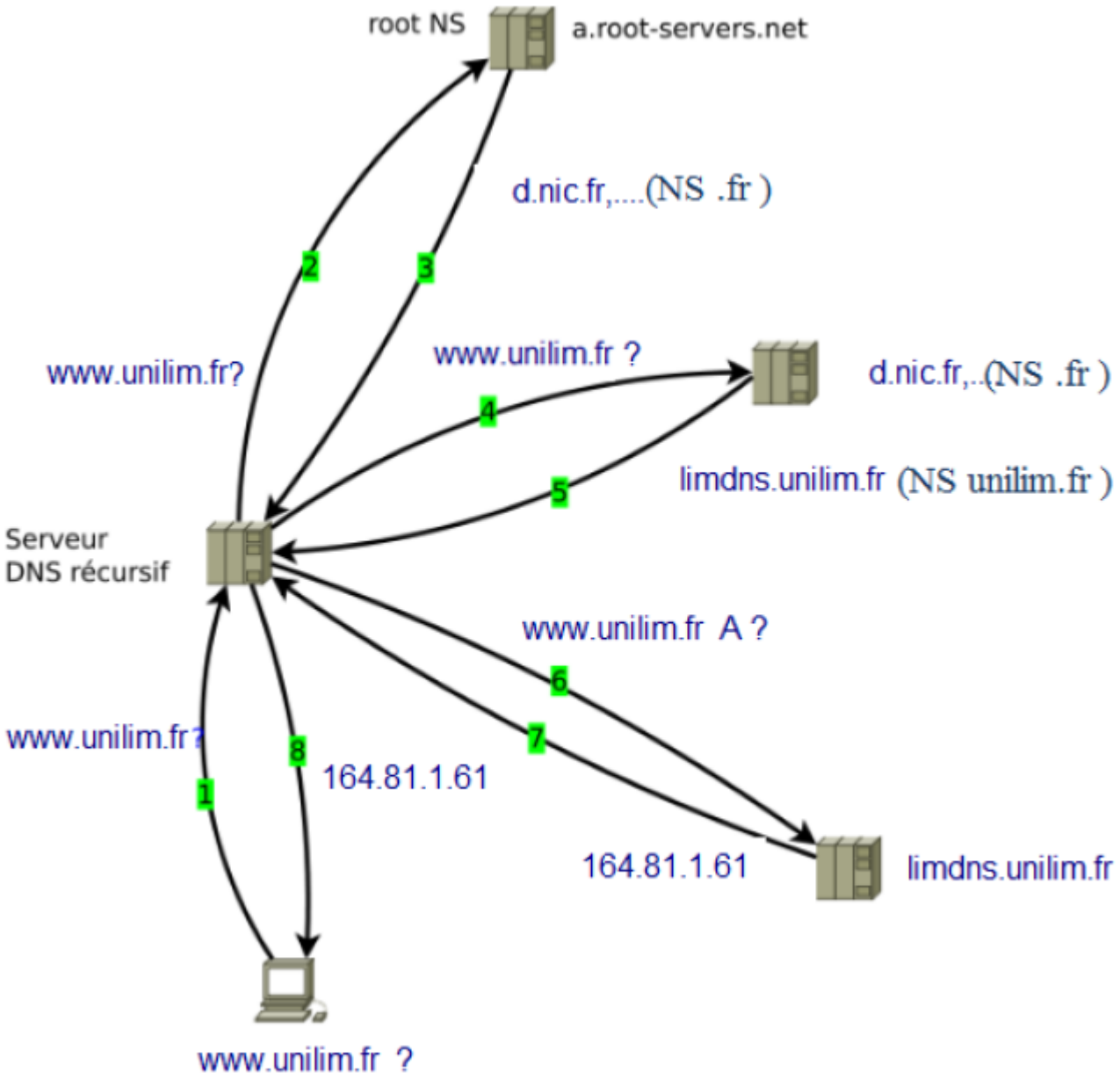

FIGURE 2 – Résolution d'un nom. D'après `https://fr.wikipedia.org/wiki/Fichier:DNS_iterations.svg`.

Ce schéma est simplifié :

- Il ne tient pas compte des caches et considère que celui du serveur récursif ne contient aucune réponse.
- Il ne tient pas compte du fait que certains serveurs faisant autorité mal configurés peuvent être récursifs et donc apporter une réponse finale au serveur récursif alors qu'ils n'auraient pas dû.

Afin d'améliorer ses temps de réponse et d'éviter de surcharger les serveurs qui font autorité sur la racine ou sur les principaux TLDs (Top Level Domain, domaines de premier niveau. Par exemple : fr., com., ...), le serveur récursif met en cache les réponses qu'il obtient. Ainsi, si un autre client l'interroge sur un nom dont il a déjà réalisé la résolution, il ne recommencera toute la résolution que si la durée de vie de la réponse a expiré.

Les serveurs récursifs ne doivent [10] pas répondre à toutes les machines, mais uniquement à un ensemble de machines bien déterminé. Par exemple : le serveur de l'université ne répond qu'aux machines qui font partie de l'université. Autre exemple : les serveurs récursifs des fournisseurs d'accès à internet français, sauf cas particulier, ne répondent qu'à leurs clients.

Les serveurs caches étant majoritairement des serveurs récursifs, nous emploierons, dans la suite de ce rapport, ces trois termes (serveur récursif / serveur cache / serveur résolveur) pour désigner un résolveur qui met en cache les informations.

Sur la plupart des réseaux domestiques, un maillon supplémentaire s'ajoute à cette liste. En effet, les routeurs (= box) fournis par les Fournisseur d'Accès à Internet (FAI) des particuliers sont équipés, sauf cas particulier, de serveurs forwarder qui se contentent de transférer les requêtes des machines du réseau local au serveur récursif du FAI.

Enfin, afin d'accroître la fiabilité du système, il est fortement recommandé d'avoir au minimum deux serveurs faisant autorité par zone. Ces serveurs faisant autorité effectuent un transfert de zone [11]. C'est-à-dire qu'un (ou plusieurs) serveur(s) maître(s), transmet(tent) la dernière version de la zone aux autres serveurs configurés en secondaires. Le transfert de zone peut-être total (AXFR) ou incrémentiel (IXFR). Pour distinguer une nouvelle zone d'une ancienne, les serveurs se basent sur le serial présent dans l'enregistrement SOA de la zone. C'est pour cette raison, entre autres, qu'il faut l'incrémenter à chaque changement, même minime, du fichier de zone.

10. Le RFC indique que les serveurs récursifs ne devraient pas être ouverts à tous les réseaux. Mais devant le nombre à peine imaginable de serveurs récursifs mal configurés, nous préférons insister et interpréter un « should not » comme un « must not ». Pour les personnes qui veulent en savoir plus : le RFC 2119 (qui fixe un sens rigoureux à des mots comme MUST, SHOULD et les négations associés) n'étant pas d'actualité lors de l'écriture des RFC relatifs au DNS, on ne peut pas s'en servir pour argumenter que les serveurs récursifs « ne devraient pas être ouverts à moins de savoir ce que l'on fait ». Tout au plus, on peut tenter de convaincre que si le RFC 2119 était antérieur, ce serait probablement le cas.

11. Ou toute autre méthode de transfert : SFTP, SCP, DRBD, réplication MySQL, réplication Active Directory, ... Mais le transfert de zone est le seul moyen de transfert standardisé dans les RFC relatifs à DNS.

2.2.3 Format des informations

Les informations publiées dans le DNS, aussi bien dans les fichiers de zone que celles récupérées auprès des serveurs faisant autorité, suivent un format normalisé, entre autres, dans les RFC 1034 et 2181.

À la base, on a un enregistrement qui associe une information (adresse IP, relais de courriel, ...) à un nom de domaine. Dans la littérature DNS, ils sont appelés Resource Record (RR). Exemple : www.unilim.fr. IN A 164.81.1.60

Un RR a un format particulier. Dans l'ordre :

- Nom de domaine (FQDN pour être précis).
- Type d'enregistrement (est-ce une adresse IPv4, le nom d'un serveur de courriel ?) [12].
- Une classe. La classe la plus utilisée sur internet est la classe IN (pour INternet). D'autres classes existent mais ne sont peu ou plus utilisées (CHaos [13], en référence à un protocole réseau qui n'a pas rencontré le succès et HeSiod [14], une alternative démodée à LDAP). Notons que la classe CH sert encore, notamment pour obtenir la version d'un logiciel serveur DNS.
- Un TTL, c'est-à-dire une durée de vie de l'information. Il désigne la durée pendant laquelle un serveur cache devrait [15] conserver l'enregistrement avant de le rafraîchir en faisant une nouvelle requête vers le serveur faisant autorité. Ce rafraîchissement n'est effectué que si le serveur cache reçoit une nouvelle demande de résolution après l'expiration du TTL.
- Le RDATA, c'est-à-dire l'information associée au FQDN. Il varie selon le type de l'enregistrement. Exemple : dans un enregistrement A, on attend une IPv4 comme RDATA.

Deux ou plus enregistrements de même type et portant sur le même FQDN forment un Resource Record Set (RRSet, « ensemble d'enregistrements » en français). Exemple :
www.unilim.fr. IN A 164.81.1.60
www.unilim.fr. IN A 164.81.1.61

12. Pour une liste des types existants, voir : `https://www.iana.org/assignments/dns-parameters/dns-parameters.xml`.
13. Voir : `https://en.wikipedia.org/wiki/Chaosnet`.
14. Voir : `https://en.wikipedia.org/wiki/Hesiod_(name_service)`.
15. Pour être complet : un serveur peut cacher la réponse "si ça lui plaît". Quand le TTL expire, ce même serveur devrait retourner chercher l'information sauf cas particulier bien compris. Notons que la remarque de la note 10 reste valable : le sens des mots employés dans les RFC n'était pas encore normalisé et ce point peut donc être sujet à discussion.

Les deux enregistrements suivants ne forment pas un RRSet car leurs types sont différents :
www.unilim.fr. IN A 164.81.1.60
www.unilim.fr. IN AAAA 2001 :dc3 : :35

Les deux enregistrements suivants ne forment pas un RRSet car leurs FQDN sont différents :
www.unilim.fr. IN A 164.81.1.60
www2.unilim.fr. IN A 164.81.1.61

Plus subtil encore : tous les RR membres d'un RRset doivent avoir le même TTL. Ils doivent, en outre, être tous transmis lors d'une demande. Un serveur faisant autorité ne doit pas transférer qu'une partie d'un RRSet et un serveur de cache ne doit pas fusionner des données d'un RRSet avec des données qu'il posséderait déjà.

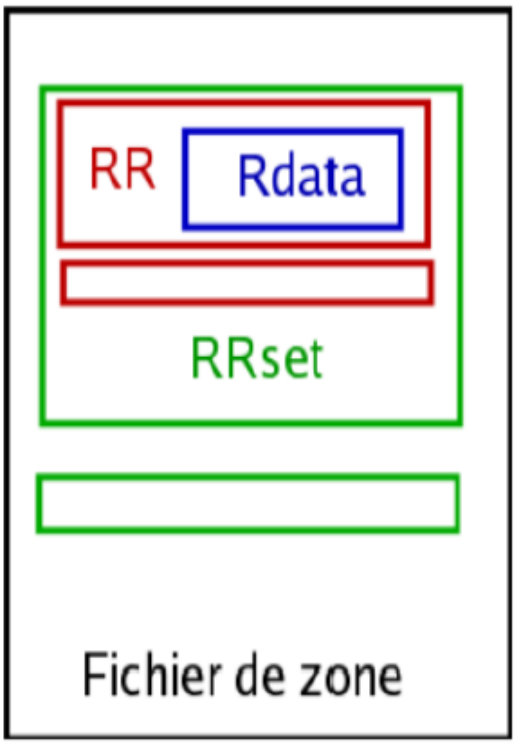

FIGURE 3 – Structure d'un fichier de zone. Source : présentation de Stéphane Bortzmeyer aux JRES 2009.

Un fichier de zone ou un message DNS est donc un ensemble de RRSet faisant partie de la même zone.

La seule exception à cette règle étant les enregistrements "glue" c'est-à-dire les enregistrements de type A ou AAAA relatifs à un enregistrement NS portant sur une délégation. Ces enregistrements sont absolument nécessaires dans le cas où le serveur faisant autorité sur une zone se trouve lui-même dans la zone. Exemple : ns.exemple.fr. fait autorité sur exemple.fr. Si un résolveur veut résoudre un nom de cette zone, par exemple www.exemple.fr., il va inter-

roger les serveurs faisant autorité sur fr. : A www .exemple.fr. ? Ces serveurs vont répondre : exemple.fr. NS ns.exemple.fr. Sans glue, le résolveur rencontrera une boucle du type :

- A ns.exemple.fr. ?
- exemple.fr. NS ns.exemple.fr.
- A ns.exemple.fr. ?
- ...

Le nom de domaine ne sera donc pas résolvable. Avec un enregistrement glue, le serveur de la zone supérieure (ici fr.) connait l'adresse du serveur de nom de exemple.fr. et il peut donc répondre au résolveur.

2.3 Mise en oeuvre

Pour bien comprendre les mécanismes du DNS, nous avons réalisé une maquette complète (comprenant un serveur faisant autorité, un serveur primaire et secondaire, un serveur récursif, un serveur forwarder ...) sous Netkit [16].

16. Pour une présentation du projet, voir le site officiel à cette adresse : `http://wiki.netkit.org/index.php/Main_Page`.

2.3.1 Présentation de notre maquette

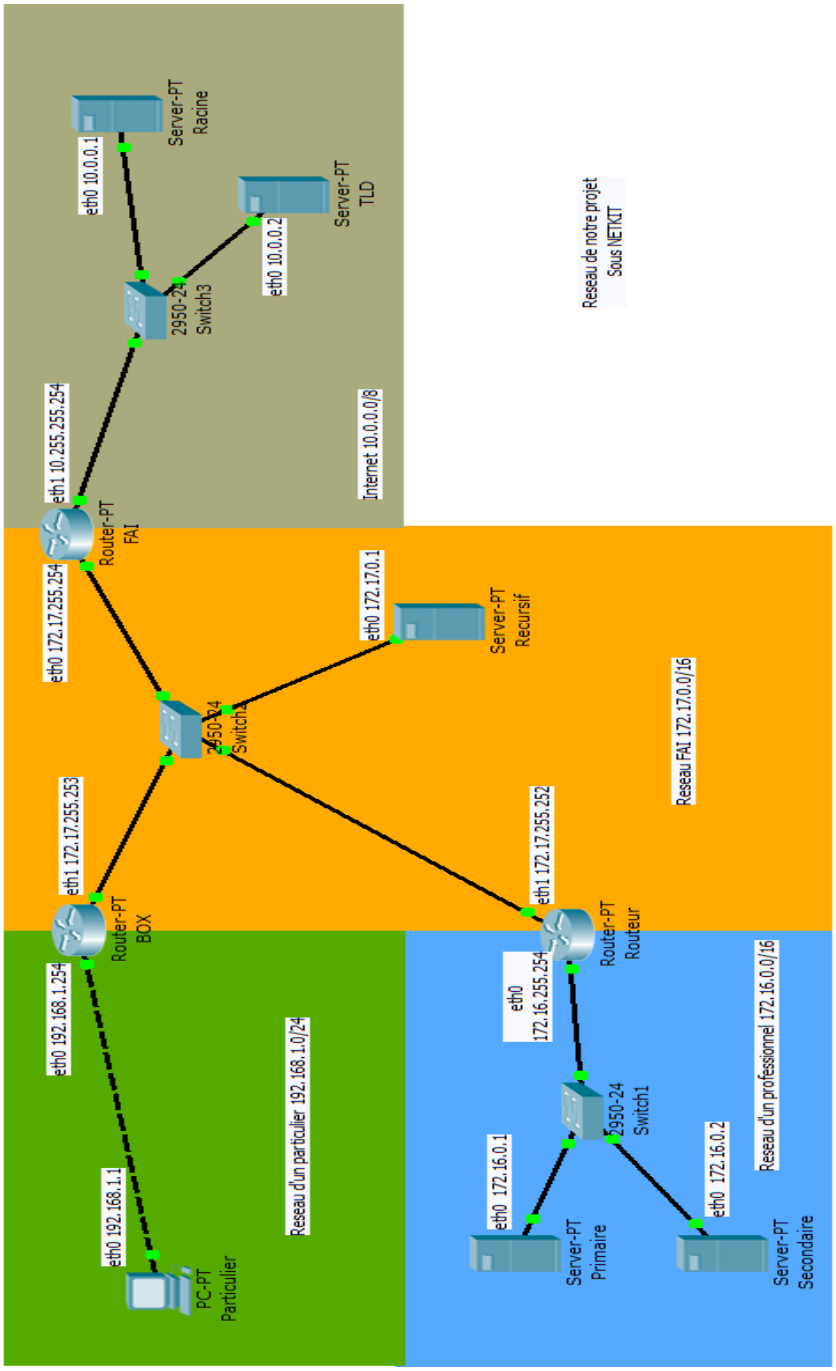

FIGURE 4 – Schéma de nos réseaux sous Netkit.

Présentation des machines :

- Le serveur Racine sera le serveur faisant autorité pour la zone « . » et pour la zone « arpa. ».
- Le serveur TLD sera le serveur faisant autorité pour un gTLD que nous avons créé : « jide. ». Il sera également le serveur faisant autorité pour la zone « in-addr.arpa. ».
- Le serveur Primaire sera le serveur primaire [17] faisant autorité pour la zone « mycorporation.jide. ».

– Le serveur Secondaire sera le serveur secondaire faisant autorité pour la zone « mycorporation.jide. ». Il y a donc un transfert de zone entre les machines Primaire et Secondaire.
– Le serveur Récursif sera un serveur récursif ouvert uniquement aux réseaux des clients du FAI (192.168.1.0/24 et 172.16.0.0/16).
– La machine Box permet de mettre en place un forwarder DNS comme on en voit dans les réseaux domestiques.
– La machine PC est un bête client DNS.

Le but de notre maquette est de résoudre la hiérarchie suivante :

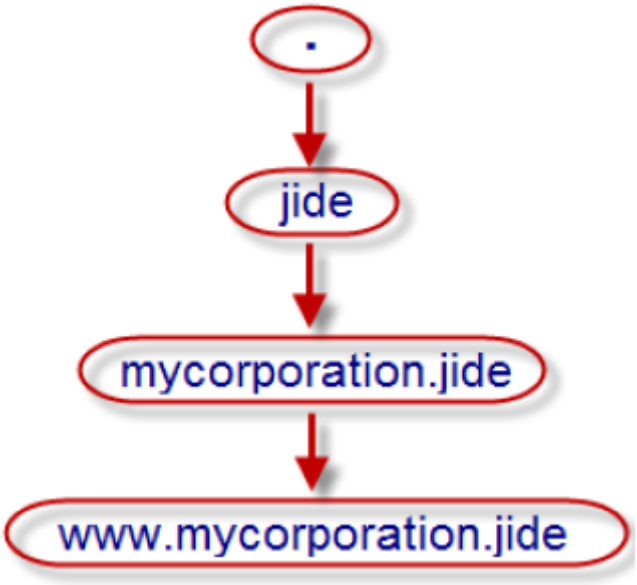

FIGURE 5 – Hiérarchie de nos noms de domaine.

Tous les serveurs utilisent BIND 9.6-ESV-R3 comme logiciel serveur. Au commencement de notre projet, nous avions prévu d'utiliser plusieurs logiciels serveur : NSD sur les serveurs Racine et TLD, BIND sur les serveurs Primaire et Secondaire, Unbound sur le serveur Récursif et Dnsmasq sur la box. Nous voulions représenter ainsi la variété des installations présentes sur Internet. Variété dont découle, en partie, la résilience actuelle du système DNS.

Néanmoins, plusieurs éléments ont fini par nous dissuader de mettre en place une telle infrastructure :

– BIND reste, malgré tout, le logiciel serveur DNS le plus documenté, ce qui est toujours appréciable.

17. Les termes primaire/secondaire sont obsolètes à cause de problèmes de cohérence, en nombre (on peut avoir plusieurs primaires) et aussi quand on a recours à des serveurs furtifs. On préfère les termes maitres/esclaves. Néanmoins, l'utilisation des termes primaire/secondaire dans ce rapport est un choix assumé.

– Le support de DNSSEC est assez différent et est loin d'être complet selon les logiciels et comme nous allons mettre en place DNSSEC [18] ...
– La configuration d'un serveur DNS est un objectif annexe de notre projet : la priorité reste les recherches concernant la sécurité du DNS.

Par ailleurs, nous avons beaucoup simplifié l'architecture réelle :
– Nous avons réduit Internet à un seul fournisseur d'accès et à un simple réseau /8 alors que, par définition, Internet est une interconnexion de plusieurs centaines milliers de réseaux et d'environ 41000 opérateurs à l'heure actuelle [19].
– Normalement, la racine et les plus gros TLD sont anycastés (redondés avec une topologie réseau particulière) [20]. Néanmoins, comme notre sujet concerne le DNS et non pas la répartition de la charge et la haute disponibilité, nous n'avons pas anycasté notre racine et notre TLD.
– Normalement, la zone in-addr.arpa. est gérée par un autre serveur qui ne gère qu'elle. Il n'y a pas de mélange entre les TLD et in-addr.arpa. Néanmoins, afin d'éviter d'avoir une machine Netkit plus ou moins inutile, nous avons regroupé in-addr.arpa. et jide. sur la même machine (TLD).
– Pour la même raison, nous n'avons pas mis deux autres serveurs pour aider notre serveur Racine et notre serveur TLD alors que les RFC (et notamment le RFC 2182) exigent qu'une zone soit servie par au moins deux serveurs faisant autorité.
– Normalement, les enregistrements de type PTR sur les IPs des serveurs qui gèrent la racine et les TLD sont déclarés par le propriétaire de la plage IP sur laquelle le serveur se trouve. Par exemple, le serveur c.root-servers.net. a pour adresse IPv4 192.33.4.12. La plage 192.33.4.0/24 appartient et est annoncée par Cogent. Donc, c'est le serveur faisant autorité sur la zone 4.33.192.in-addr.arpa. et appartenant à Cogent qui doit posséder l'enregistrement PTR de 12.4.33.192.in-addr.arpa. C'est effectivement le cas puisque les serveurs faisant autorité sur la zone 4.33.192.in-addr.arpa. sont authX.dns.cogentco.com. (remplacez X par un chiffre). Comme sur notre maquette, nous avons réduit Internet à un simple réseau /8 et pour éviter l'utilisation d'une machine Netkit supplémentaire, c'est le serveur TLD, dans la zone in-addr.arpa. qui déclare son PTR ainsi que celui de la racine.

18. Cette remarque concerne surtout dnscache et dnsmasq que nous voulions employer, en partie, dans notre maquette, au début.
19. Sources : Benjamin BAYART - conférence « DIY ISP : devenez votre propre fournisseur d'accès à Internet »- Pas sages en Seine 2010. On peut aussi regarder le nombre de numéros d'Autonomous System qui s'échangent actuellement des préfixes sur Internet grâce à des moniteurs comme bgpmon.net.
20. Pour une définition plus complète, voir : `https://fr.wikipedia.org/wiki/Anycast` et `http://blog.spyou.org/wordpress-mu/2011/01/11/cest-quoi-lanycast/`.

– Comme nous l'avons expliqué plus haut, tous nos serveurs utilisent BIND comme logiciel serveur, ce qui est une simplification de la réalité : rien que la racine officielle de l'ICANN utilise au moins deux logiciels serveurs : BIND et NSD.

2.3.2 Directives de configuration et sécurité

La sécurité d'un serveur dépend de sa bonne configuration. Nous avons décidé d'écraser le named.conf fourni par défaut et d'écrire le nôtre "from scratch".

Nous allons donc faire une liste, non exhaustive, des directives de BIND nécessaires pour chaque type de serveur DNS. Elles sont à mettre dans le fichier named.conf.

Certaines directives peuvent être mises dans le bloc « options » et s'appliquer ainsi à toutes les zones gérées par le serveur ou bien dans la définition de chaque zone. Nous avons préféré suivre l'approche "tout bloquer dans les options et autoriser ensuite en fonction des zones".

Nous n'avons pas utilisé les vues. Nous avons concentré toute la configuration dans le même named.conf alors que la tendance actuelle est de séparer les options des zones dans différents fichiers.

Les fichiers de zones ainsi que les fichiers de configuration complets de nos serveurs sont annexés à ce rapport.

Pour tous les serveurs : Il convient de masquer la version de BIND utilisée. En effet, tout le monde peut la récupérer auprès du serveur avec une requête du type « dig @serveur version.bind chaos txt ». Notons bien que le masquage de la version n'est pas une sécurité. Pour la masquer, il faut placer la directive suivante dans le bloc « options » : version « message » ;

Il faut également limiter les interfaces réseau d'écoute. Cela se fait en plaçant la directive suivante dans le bloc « options » : listen-on {ip1 ; ip2 ;} ;

L'équivalent pour IPv6 est listen-on-v6.

Pour un serveur faisant autorité sans secondaire : On commence par autoriser tous les clients à interroger notre serveur en ajoutant la directive suivante dans le bloc « options » : allow-query { any ; } ;

Comme nous l'avons déjà dit, un serveur faisant autorité est itératif, il doit donc refuser toutes les requêtes récursives. Cela se fait avec la directive suivante qui doit être placée dans le bloc « options » : recursion no ;

Mais cela ne suffit pas : dans le cas où des données sont en cache, un intrus peut malgré tout les récupérer. Cela est surtout vrai sur le serveur récursif. Pour contrer cela, il faut ajouter l'option suivante dans le bloc « options » : allow-query-cache { none ; } ;

On interdit au serveur de communiquer des informations additionnelles concernant une autre zone. Cela s'avère utile pour contrer une variante de l'attaque par réflexion/amplification apparue début 2009 et qui consiste à demander les NS de la racine. Une directive « recursion no » ne

suffit pas. On interdit également au serveur de suivre des enregistrements CNAME/DNAME qui pointent sur une autre zone. Cela se fait en ajoutant les directives suivantes dans le bloc « options » : additional-from-auth no ; additional-from-cache no ;

Puisque ce serveur est le seul à faire autorité sur la zone, il faut interdire le transfert de zone (on peut tenter un transfert avec la commande dig @serveur axfr <zone>). Cela se fait en ajoutant la directive suivante dans le bloc « options » : allow-transfer { none ;} ;

Des indésirables peuvent néanmoins forger de fausses requêtes de notification et mener ainsi une attaque DoS [21]. Pour contrer cela, il faut ajouter l'option suivante dans le bloc « options » : notify no [22] ;

Pour un serveur faisant autorité avec secondaire(s) : Il suffit de reprendre le paragraphe précédent. La seule différence concernera le transfert de zone. Il sera toujours interdit dans le bloc « options » mais autorisé dans les sections des zones qu'on veut pouvoir transférer. Il ne sera autorisé que pour certains serveurs identifiés soit par une adresse IP, soit, et c'est encore mieux, par une authentification avec le mécanisme TSIG/TKEY. TKEY rend les mêmes services que TSIG mais évite le transfert manuel de la clé entre les deux fichiers de configuration en se basant sur Diffie-Hellman.

Dans notre cas, nous autoriserons le transfert des zones mycorporation.jide. et 16.172.in-addr.arpa. entre les serveurs Primaire et Secondaire en utilisant TSIG.

Dans les sections « zone "mycorporation.jide" IN » et « zone "16.172.in-addr.arpa" IN » des deux named.conf (celui du serveur primaire et celui du serveur secondaire), nous ajouterons la directive suivante : allow-transfer{ 172.16.0.2 ; } ; Elle autorise le transfert de ces deux zones pour 172.16.0.2 (IP de Secondaire).

On génère une clé TSIG avec la commande « dnssec-keygen -a HMAC-sha256 -b 256 -n HOST cle » [23].

On crée un fichier tsig.conf. On définit son propriétaire comme étant root et son groupe comme étant bind. On donne les permissions 550 à ce fichier.

21. Consiste à envoyer des requêtes de notification à un/des secondaire(s) depuis n'importe quelle machine afin de causer des requêtes de transfert intempestives. Pour empêcher cela, les notifications utilisent aussi TSIG et/ou les ACL par IP.

22. Notons que cette directive n'est pas nécessaire ici puisqu'un serveur primaire ne reçoit pas de notifications mais en envoi. Cette directive bloque cependant l'envoi de notification. Elle est simplement mise ici pour faire un gabarit et ne pas l'oublier au moment venu.

23. Dans un vrai déploiement, l'algorithme et la taille de la clé devront être choisis en fonction de l'importance de la zone et du nombre de réseaux qui séparent le serveur primaire de son secondaire.

On édite ce fichier pour qu'il contienne :

```
key "TRANSFERT" {
        algorithm hmac-sha256;
        secret "fVHHGLEwSJTRUIa6PgIhJmYz4Vcim11QlgInnCUfEIw=";
};

server 172.16.0.2 {
        keys {
                TRANSFERT;
        };
};
```

L'algorithme et le secret sont récupérés depuis le fichier cle.key généré par dnssec-keygen. On supprime ensuite les fichiers générés par dnssec-keygen.

On édite à nouveau le named.conf du primaire pour y ajouter la directive : include "/etc/bind/tsig.conf" ;

Sur le secondaire uniquement, dans les sections qui définissent les zones, il faudra ajouter la directive : masters { 172.16.0.1 ; } ;

Dans le même temps, toujours dans le named.conf du secondaire, on autorisera le serveur primaire à notifier notre secondaire des mises à jour des zones en ajoutant la directive suivante dans les sections qui définissent les zones : allow-notify { 172.16.0.1 ; } ; . Dans le named.conf du primaire, on passera la directive « notify » à « yes » afin que notre primaire informe notre secondaire des changements intervenus sur les zones. Notons que l'instruction notify, quand elle est positionnée à « yes », ne notifie que les serveurs définis dans les RRset NS d'une zone ou ceux spécifiés en plus dans la directive also-notify.

Sur le secondaire, on inclura également le fichier tsig.conf comme précédemment, mais celui contiendra l'IP de Primaire

On demande donc à notre secondaire de contacter 172.16.0.1 en utiliser la clé TRANSFERT pour procéder à un transfert des zones mycorporation.jide. et 16.172.in-addr.arpa.

Pour un serveur récursif : Un serveur récursif doit proposer ses services à un ensemble de machines bien identifié. On limite donc les réseaux qui peuvent nous interroger : le réseau du particulier, le réseau du professionnel, le réseau du FAI et l'adresse locale avec les directives allow-query, allow-recursion, allow-query-cache.

On désactive le transfert de zone et la notification comme précédemment.

Pour un serveur forwarder : On indique le serveur récursif auquel demander la résolution des noms avec la directive suivante placée dans le bloc « options » : forwarders { 172.17.0.1 ; } ;

On indique au serveur qu'il ne doit jamais tenter de résoudre les requêtes même si le serveur récursif ne lui répond pas en plaçant la directive suivante dans le bloc « options » : forward only ;

On limite encore une fois l'accès au serveur aux machines du réseau du particulier à l'aide des directives allow-query, allow-recursion, allow-query-cache.

Comme sur les autres serveurs, on désactive encore le transfert de zone et la notification (même si c'est assez inutile de le faire).

2.3.3 Validation de notre maquette

Nous avons vérifié les configurations de nos serveurs faisant autorité à l'aide de Zonecheck. Ce logiciel est préconisé, entre autres, par l'AFNIC. Il effectue, de manière automatique, un ensemble important de tests qu'un humain pourrait faire manuellement avec l'utilitaire dig mais qu'il aurait très peu de chance de mener à terme correctement (c'est-à-dire avec la certitude ne n'avoir commis aucun oubli/erreur).

Zonecheck a validé l'intégralité de nos configurations. En effet, les erreurs qu'il nous a remontées sont sans importance et dépendent des contraintes de notre maquette. En effet, Zonecheck nous signale :

- Que nos serveurs faisant autorité sont sur le même Autonomous System. Ce qui est normal dans notre maquette.
- Que nos serveurs faisant autorité ont des IP RFC 1918. Ce qui est normal dans notre maquette.
- Que nous n'avons pas deux serveurs faisant autorité pour toutes nos zones (et notamment pour « . », « arpa. », « jide. » et « in-addr.arpa. » alors que c'est obligatoire au sens des RFC. Ce choix est volontaire afin de limiter le nombre de machines Netkit.
- Que nous n'avons pas de MX dans notre racine et notre TLD. Il n'y a pas non plus de MX dans la racine officielle de l'ICANN ni dans les gTLDs (nous avons vérifié pour .fr, .net et .info). Cela s'explique par le fait que Zonecheck a été développé pour permettre le test de zones "classiques" (Second Level Domain ou en dessous), situées en dessous des TLDs et pour lesquelles un enregistrement MX est obligatoire.
- Que nous avons, dans l'enregistrement SOA de notre racine, une valeur de rafraîchissement trop petite. Là encore, nous avons la même valeur que dans la racine officielle de l'ICANN. Cela s'explique, là encore, par le fait que Zonecheck a été développé pour permettre le test de SLD [24].

De plus, comme les fichiers de zones de la racine et d'arpa. sont mis librement à disposition par l'InterNIC et l'ICANN, nous avons pu comparer manuellement nos zones racine et arpa. à celles de la racine officielle de l'ICANN.

24. Notons que Zonecheck est paramétrable et qu'il doit être possible de désactiver les tests impropres pour un TLD.

3 DNSSEC

3.1 Présentation succincte

DNSSEC (Domain Name System Security Extensions) n'est pas un nouveau protocole mais une extension de DNS. Ainsi, les principes de fonctionnement de DNS sont préservés et la transition est facilitée. Cela signifie également que les machines qui traitent seulement DNS, peuvent transmettre les enregistrements sécurisés sans aucune intervention humaine. Il y a donc une compatibilité ascendante. Les logiciels que nous connaissons (Firefox, OpenSSH, Thunderbird, ...), n'ont pas à être adaptés ni modifiés pour y ajouter un quelconque support de DNSSEC.

Le but de DNSSEC est d'authentifier les données distribuées par le DNS à l'aide de la cryptographie asymétrique. Comme la confidentialité des données importe peu (les données contenues dans le DNS sont déjà majoritairement publiques), on se contente de signer les enregistrements. Chaque enregistrement est donc condensé avec une fonction de hachage puis le condensat est signé. Néanmoins, le canal de communication n'est pas sécurisé par DNSSEC et ce n'est pas son rôle. En effet, on veut s'assurer que la réponse et son origine initiale sont correctes, pas que les intermédiaires sur le chemin sont tous des serveurs "bien intentionnés" et non corrompus.

La meilleure analogie est celle d'une lettre postale. Authentifier les données est l'équivalent d'une lettre postale signée et scellée. Cela permet de garantir que, quel que soit le chemin emprunté par la lettre, elle est authentique. C'est le rôle de DNSSEC. Authentifier le canal serait l'équivalent d'identifier le messager et donc de vérifier la carte professionnelle du facteur qui vous apporte la lettre pour vérifier que sa profession est bien facteur. Ce point-là ne fait pas partie du cahier des charges de DNSSEC.

DNSSEC est évolutif. En effet, il ne dépend pas d'un algorithme cryptographique particulier : ce n'est qu'un paramètre. Si demain les progrès de la cryptanalyse permettent de casser un algorithme massivement utilisé de nos jours (comme RSA), il suffira d'en changer. La gêne sera donc temporaire et ne nécessitera pas de revoir entièrement DNSSEC.

3.2 Présentation détaillée

DNSSEC n'est pas une idée nouvelle. En effet, dès le début des années 1990, des acteurs se rendent compte des faiblesses de DNS et des discussions informelles sont même engagées au sein de l'IETF au milieu des années 1990. En 1999, le RFC 2535 est même publié. L'IETF se rendra compte, dès 2001, que la mise en place sera difficile, voire impossible, et décidera de retravailler le RFC. En 2005, les trois RFC (4033, 4034, 4035) décrivant DNSSEC sont publiés. En 2007, un premier domaine de premier niveau, .se est signé. En 2010, la racine DNS officielle de l'ICANN est signée ainsi que quelques domaines de premier niveau.

Les logiciels n'ont pas besoin d'être adaptés pour bénéficier de DNSSEC. Ils peuvent l'être si l'on souhaite qu'ils réalisent eux-mêmes la validation des signatures cryptographiques mais cela est assez contre-productif, nous en reparlons plus loin dans ce rapport. Si le résolveur qu'utilise le client est configuré pour utiliser DNSSEC et valider les signatures, alors toutes les applications du client profiteront automatiquement des apports de DNSSEC. Les logiciels émettront une requête DNS classique, sans demander la prise en charge de DNSSEC. Le résolveur qu'utilise le client résoudra le nom demandé en activant le support de DNSSEC et validera, s'il les reçoit, les signatures. Il transmettra le résultat au client en occultant les champs propres à DNSSEC. Sous condition que les données n'ont pas été modifiées entre le résolveur local et le client (nous y reviendrons), ce dernier aura la garantie que la réponse est exacte, même dans le cas d'un nom qui n'existe pas.

3.2.1 Gestion des clés

Comme DNSSEC repose sur la cryptographie asymétrique, nous avons besoin d'au moins une paire de clés (clé publique – clé privée) par zone à signer. Les paires de clés sont donc associées à une zone et non à un serveur. Si un même serveur distribue plusieurs zones, il manipulera plusieurs paires de clés.

Néanmoins, DNSSEC n'est pas conseillé pour reposer sur l'utilisation d'une seule paire de clés : l'utilisation de deux paires de clés par zone est préférée et préférable. En effet, une seule paire de clés présente des risques de sécurité. D'une part, la même clé sert à tout signer. Cela fait donc un échantillon plus grand pour les cryptanalyses. Et plus on a un échantillon important à analyser, plus les chances de succès augmentent (bien que ce ne soit pas le seul critère dont dépend la sécurité). On peut résoudre ce problème en changeant fréquemment la clé mais cela reste une opération pénible (surtout qu'il faut indiquer à la zone mère qu'on a changé de clé, nous y reviendrons) et on n'exclut pas le risque que le changement de clé ne soit pas effectué de ce fait ou que des erreurs de manipulations entraînent des erreurs de validation. En utilisant deux paires de clés, le contenu signé par chacune diminue et l'on réduit donc le matériel mis à la disposition des cryptanalyses. De plus, on simplifie le processus de roulement des clés (car le changement de la clé qui signe la zone n'est pas à signaler à la zone mère, nous y reviendrons) et on augmente donc les chances que ce roulement soit effectué par le plus grand nombre.

La première clé, dite ZSK (Zone Signing Key) signe tous les enregistrements de la zone, à l'exception du RRset DNSKEY (qui contient les clés utilisées par la zone ou qui sont amenées à l'être). Il s'agit d'une clé de taille réduite. La deuxième clé, dite KSK (Key Signing Key), signe uniquement le RRset DNSKEY et sert donc de maillon de confiance. Sa taille sera donc plus longue.

Les documents officiels indiquent que les KSK de la racine de l'ICANN ont une longueur de 2048 bits alors que les ZSK de la racine ont une longueur de 1024 bits. Il est recommandé d'adapter la taille des clés en fonction de l'importance de la zone sans aller en dessous de 1024 bits. Il convient également d'adapter la clé en fonction des intermédiaires : mettre une KSK 4096 bits sur secure.unilim.fr. ne sert presque à rien puisque les clés de la racine sont plus faibles (mais mieux protéger en contrepartie). Si l'on veut vraiment une zone sécurisée, spécifique et réservée, on choisira une clé forte que l'on diffusera chez les utilisateurs sans passer par la hiérarchie DNS, bien que cette solution ne passe pas à l'échelle.

Afin de garantir la sécurité, il convient de procéder à un roulement régulier des clés. Les documents officiels annoncent une durée de vie de deux à cinq ans pour la KSK de la racine de l'ICANN et une durée de vie de treize mois pour la ZSK correspondante. Pour les zones "normales", il est recommandé de choisir une durée de vie de douze mois pour la KSK et d'un mois pour la ZSK. Néanmoins, il convient de noter que les durées de vie optimales sont encore en discussion et peuvent varier selon l'importance de la zone mais il est évident qu'une clé plus courte doit être changée plus souvent qu'une clé plus longue.

Il faut noter que, dans DNSSEC, les clés n'expirent pas toutes seules car elles n'ont pas une durée de vie intrinsèque. Seules les signatures expirent. Un processus de renouvellement des clés doit donc être prévu.

Le roulement des clés est certainement le processus le plus critique de DNSSEC car la moindre erreur peut entraîner des erreurs de validation. Même le RFC 4641, censé donner les bonnes pratiques opérationnelles, contient des erreurs dans les tableaux de roulement des clés ! Pour simplifier et automatiser le processus, des logiciels existent comme OpenDNSSEC [25]. Leur seul but est d'automatiser la signature des zones et le roulement des clés.

Il est à noter qu'il existe deux types de roulement de clés : le roulement périodique et le roulement d'urgence.

Il est évident que le deuxième type est impossible à préparer. Il faut alors faire un compromis entre la rupture de la chaîne de confiance dans le cas d'un changement immédiat des clés et un risque d'attaque si l'on conserve trop longtemps les clés compromises. La politique adoptée sera spécifique et locale. Par exemple : il est possible de publier, en permanence, une clé supplémentaire dans le RRset DNSKEY de la zone. Cette clé ne servira qu'en cas de problèmes.

Pour le premier type de roulement, il est nécessaire de le prévoir dans une procédure. Ce qui va poser le plus de problèmes, c'est le fait que la résolution d'un nom de domaine met en cache les informations à plusieurs niveaux et qu'il faudra donc prendre en compte ces temps de mise à jour des informations pour faire en sorte que la zone ne soit jamais invalide.

D'autres paramètres dépendant de la configuration locale doivent être aussi pris en compte. Par exemple : si la zone est distribuée par plusieurs serveurs faisant autorité, il faudra prendre en compte la durée de rafraîchissement de la zone sur les serveurs secondaires. En effet, un client pourrait très bien s'adressait à un des serveurs secondaires d'une zone et obtenir des informations dépassées vis-à-vis du serveur primaire. Et si en plus le client qui obtient ces informations dépassées est un serveur récursif qui cache les données [26] ? Autre exemple : et si

25. Pour une présentation d'OpenDNSSEC, voir : `http://www.bortzmeyer.org/opendnssec-debut.html`.

la clé était utilisée comme ancre de confiance par des résolveurs, comment les prévenir ? Pour simplifier, nous ne tiendrons pas compte de ces variables dans la suite de notre raisonnement.

La bonne démarche est de raisonner de la manière suivante : lorsque de nouvelles données apparaissent, il doit être possible d'utiliser les signatures du cache et lorsque d'anciennes données sont présentes dans le cache, il doit être possible d'utiliser les nouvelles signatures. Il faut donc s'assurer soit que toutes les clés (anciennes et nouvelles) soient disponibles dans la zone, soit que toutes les signatures (anciennes et nouvelles) soient disponibles dans la zone. La zone doit être validable à tout instant.

Le premier cas s'appelle schéma de prépublication et est adapté au renouvellement de la ZSK et de la KSK. Le deuxième s'appelle schéma de double signature et n'est pas adapté au renouvellement de la ZSK. Il peut néanmoins être utilisé pour le renouvellement de la KSK. La méthode de pré-publication demande plus d'efforts mais la méthode de double signature produit un fichier de zone trop lourd (car on obtient deux signatures par RRset). La méthode de double signature est envisageable pour le renouvellement de la KSK car il faudra produire une double signature uniquement pour le DNSKEY RRset. Aussi bien pour la KSK que pour la ZSK, la procédure recommandée est celle de la prépublication.

Pour la ZSK, si l'on suit la procédure de prépublication, le roulement se fera en trois étapes :

1. On introduit une nouvelle ZSK dans le fichier de zone. Elle est passive et ne signe rien.
2. La nouvelle ZSK signe les enregistrements. L'ancienne reste dans le fichier de zone pour permettre aux résolveurs qui ont récupéré les signatures juste avant le début de cette phase de continuer à valider les signatures.
3. La nouvelle ZSK signe les enregistrements et l'ancienne ZSK est supprimée du fichier de zone.

Pour déterminer la durée des phases 1 et 2, il faut prendre en compte :

– Le TTL du DSNKEY RRset de la zone ainsi qu'une évaluation du TLL réellement appliqué puisque les résolveurs sont souvent configurés (à tort) avec des TTL différents du TTL original. Souvent, on voit le TTL forcé à 1 jour.
– L'intervalle entre deux signatures de la zone.
– La durée de vie des signatures de la zone.

26. Si l'on interroge un serveur secondaire, les informations obtenues en retour peuvent être dépassées vis-à-vis du primaire (car ce dernier n'a pas notifié le secondaire d'une mise à jour et le délai avant rafraîchissement de la part du secondaire n'est pas encore écoulé) mais doivent rester validables. Si l'on ajoute un serveur récursif dans la chaîne et que c'est lui qui interroge le serveur secondaire d'une zone, on obtient un temps t = temps de rafraîchissement de la zone + TTL de l'information. Il faudra prendre en compte ce temps supplémentaire.

Exemple : on suppose qu'une zone signée tous les jours et dont le TTL du DNSKEY RRset est de deux jours. On veut faire rouler la ZSK toutes les semaines. Chaque ZSK restera donc onze jours dans le fichier de zone dont sept jours où elle signera la zone. La nouvelle ZSK sera publiée deux jours avant la date du roulement et l'ancienne ZSK sera supprimée du fichier de zone deux jours après le roulement.

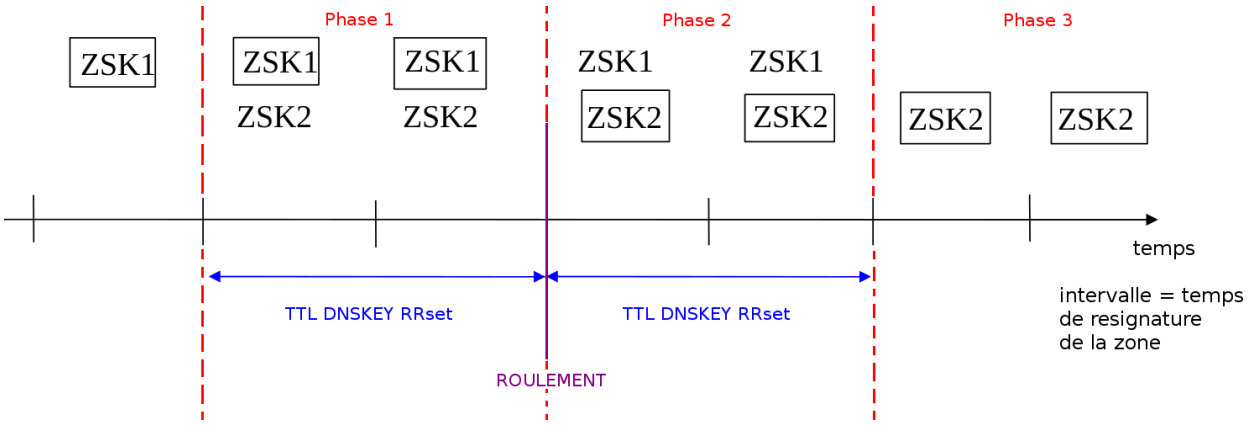

FIGURE 6 – Exemple de roulement d'une ZSK. D'après la présentation de Stéphane Bortzmeyer aux JRES 2009.

Pour la KSK, on applique la même procédure que pour la ZSK. Néanmoins, il faut prendre en compte le fait qu'un enregistrement DS situé dans la zone parente pointe sur la KSK de la zone fille afin de déléguer la confiance. Il ne faut pas rompre la chaîne de confiance et il faut donc propager le DS de la même manière que les DNSKEY RRset. Là encore, deux procédures sont possibles : publication de deux DS ou utilisation de deux KSK par la zone fille. Il faut tenir compte du délai de signature de la zone parente.

Si l'on utilise à nouveau la méthode recommandée de pré-publication, on retrouve nos trois étapes :

1. On introduit une nouvelle KSK dans le fichier de zone. Elle est passive et ne signe rien. On communique également notre nouveau DS à la zone fille pour qu'il soit incorporé dans le fichier de zone.
2. La nouvelle KSK signe les enregistrements (RRset DNSKEY). L'ancienne reste dans le fichier de zone pour permettre aux résolveurs qui ont récupéré les signatures juste avant le début de cette phase de continuer à valider les signatures.
3. La nouvelle KSK signe les enregistrements et l'ancienne KSK est supprimée du fichier de zone. On communique également avec la zone parente pour qu'elle retire l'ancien enregistrement DS.

Exemple : on suppose une zone parente et une zone fille signées tous les jours mais pas forcément en même temps car cela a peu de chances de se produire en pratique. Le TTL du DNSKEY RRset est toujours de deux jours. Le TTL du DS est d'un jour. Il faut prévoir une phase deux plus longue à cause de la nécessaire synchronisation KSK/DS.

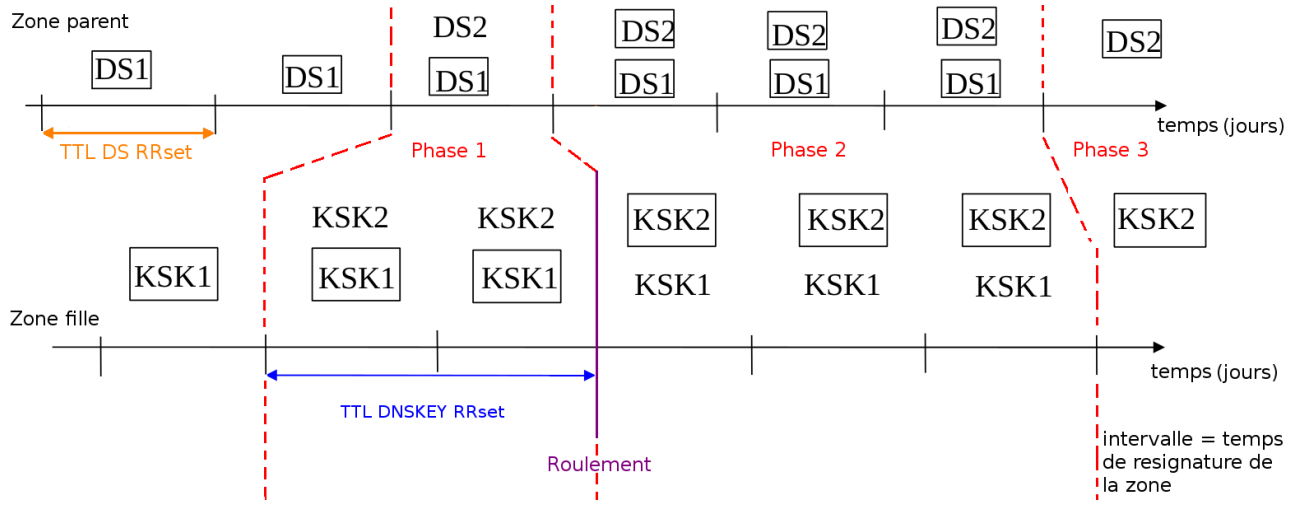

FIGURE 7 – Exemple de roulement d'une KSK. D'après la présentation de Stéphane Bortzmeyer aux JRES 2009.

Un petit mot concernant le stockage des clés. Il n'est pas spécifique à DNSSEC mais à tout système reposant sur la cryptographie et c'est pour cela que nous ne nous attarderons pas sur ce point.

Le stockage des parties privées des KSK et des ZSK doit être sûr. C'est-à-dire qu'il faut empêcher des tiers de se procurer les clés sans autorisation (car le tiers pourra alors signer de faux enregistrements) mais qu'il faut également permettre leur récupération en cas de problème (sinon, on ne peut plus signer la zone). Les deux objectifs étant contradictoires (c'est-à-dire : l'objectif de récupération impose de dupliquer les clés sur plusieurs sites alors que cela amplifie les possibilités d'une récupération malveillante), il faut faire un compromis.

On peut opter pour plusieurs solutions allant d'un dossier dont le propriétaire et le groupe sont bien choisis et ayant des droits 400 à un HSM (Hardware Security Module [27]) en passant par un softHSM (HSM logiciel).

Pour plus de sécurité, on peut également envisager de signer hors ligne [28] en utilisant une machine (virtuelle ou non) uniquement pour l'opération de signature qui alimenterait un serveur primaire furtif qui lui même alimenterait les serveurs en production. Le tout avec de bonnes règles sur les pare-feux pour éviter les remontées vers le serveur primaire furtif. La machine dédiée à la signature ne serait allumée que durant l'opération de signature et serait maintenue

27. Pour une définition, voir : `https://en.wikipedia.org/wiki/Hardware_Security_Module`.

saine. Si l'on utilise la fonctionnalité de mise à jour dynamique d'une zone, on ne peut signer qu'en ligne [28] avec les risques d'exposition des clés privées que cela comporte. L'architecture précédemment décrite devient alors fortement conseillée.

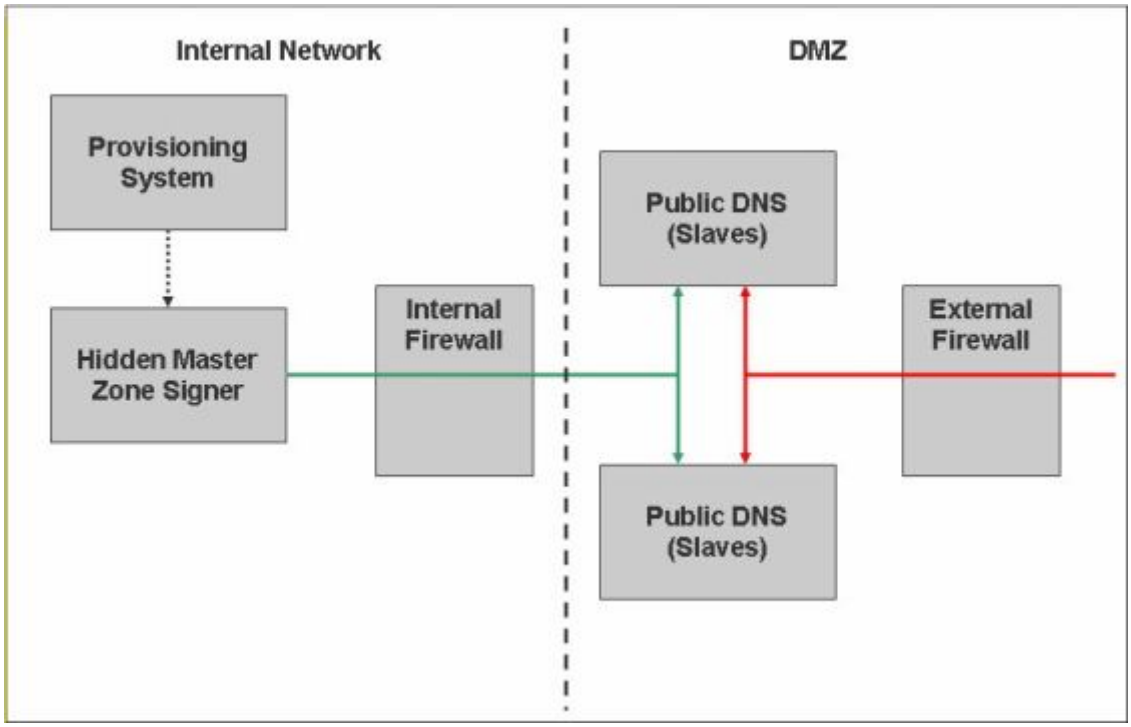

FIGURE 8 – Schéma d'une architecture de signature en entreprise. Source : `http://www.zytrax.com/books/-dns/info/choose-dnssec.html`.

28. Signer hors ligne signifie qu'une zone est signée à un instant précis, sur une machine déconnectée du réseau et non pas lors de la réception d'une requête. Les clés privées ne sont donc à aucun moment sur le(s) serveur(s) faisant autorité sur la zone car elles y sont inutiles. Dans le cas d'une signature en ligne, typiquement lors de l'utilisation de la fonctionnalité de mise à jour dynamique d'une zone (DDNS), le serveur doit avoir connaissance des clés privées pour signer le nouvel enregistrement lors de sa soumission.

3.2.2 Nouveaux enregistrements

L'idée de DNSSEC est de distribuer les clés et les signatures dans le DNS lui-même. Une des directives du cahier des charges de DNSSEC est qu'il ne doit pas changer le protocole DNS : il doit permettre l'utilisation de l'infrastructure existante. Ainsi, DNSSEC peut fonctionner à travers des serveurs non modifiés. Ainsi, si des intermédiaires sur le chemin ne sont pas en mesure de traiter DNSSEC, ils laissent quand même passer la réponse. Exemple concret : les résolveurs du réseau Unilim ne font pas de validation DNSSEC mais laissent passer les questions/réponses des clients qui souhaitent effectuer la validation eux-mêmes.

L'idée est donc d'ajouter de nouveaux types d'enregistrement dédiés à la diffusion des signatures et aux clés, en conservant le format normalisé des autres types d'enregistrement DNS.

Ainsi, DNSSEC introduit au moins sept nouveaux types d'enregistrement principaux. Trois d'entre eux seront détaillés dans la partie « Non-existence d'un enregistrement ».

À l'exception des enregistrements de types DNSKEY, DS et DLV, les autres enregistrements seront insérés automatiquement par le signeur, c'est-à-dire par le logiciel qui signera la zone.

L'enregistrement DNSKEY (DNS public KEY) permet de stocker les parties publiques des clés utilisées pour signer la zone. Il y a donc deux enregistrements DNSKEY par zone : un pour la KSK, un pour la ZSK.

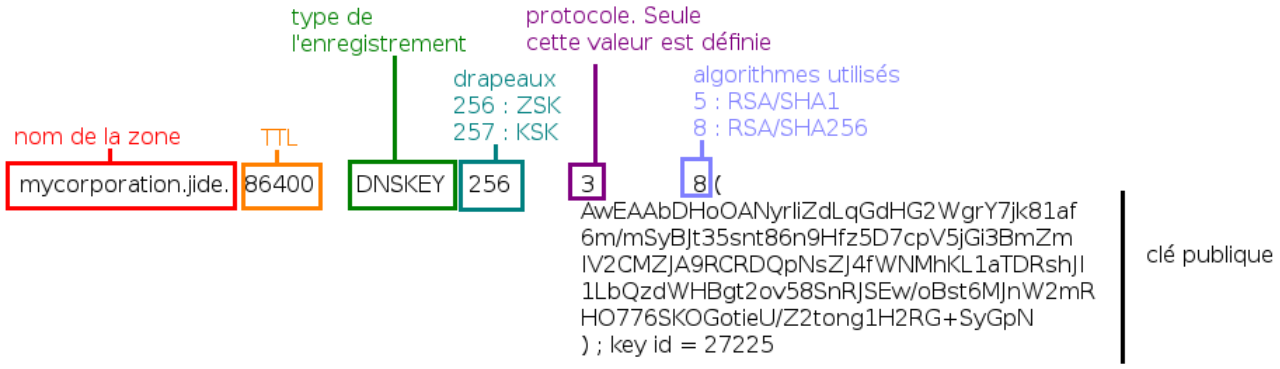

FIGURE 9 – Structure d'un enregistrement DNSKEY.

Les algorithmes utilisables ainsi que le numéro associé sont définis dans l'annexe du RFC 4034 et dans le RFC 5702 relatif à l'utilisation de la famille SHA2 dans DNSSEC.

L'enregistrement RRSIG (Resource Record SIGnature) permet de stocker la signature de chaque RRset appartenant à la zone à l'exception des RRSIG eux-mêmes (car la signature contenue dans un RRSIG s'applique déjà à l'enregistrement couvert par le RRSIG ainsi qu'aux données du RRSIG, à l'exception du champ signature). Le critère d'appartenance à la zone signifie que les enregistrements "glue" ne seront pas signés. Néanmoins, les enregistrements DS (nous y reviendrons) seront signés.

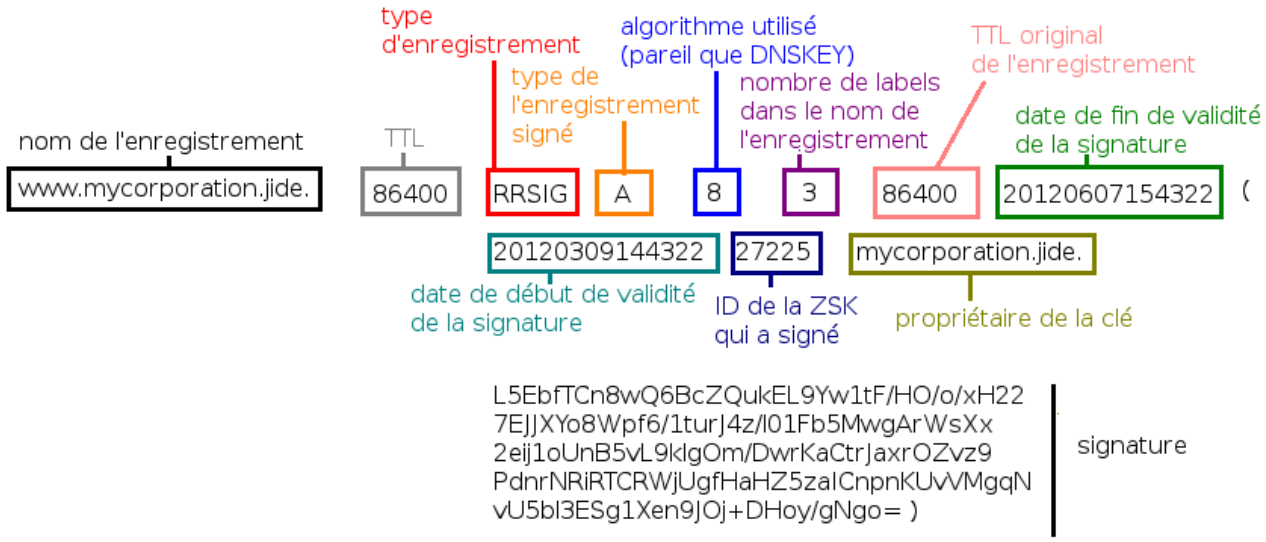

FIGURE 10 – Structure d'un enregistrement RRSIG.

Le champ « label » indique le nombre d'étiquettes que comporte le nom associé à l'enregistrement RRSIG. Cette indication permet de vérifier si la réponse a été générée à partir d'un wildcard (joker) [29]. Pour rappel, un joker permet de définir une réponse par défaut pour tous les sous-labels qui n'existent pas. Avec DNSSEC, c'est la valeur du champ Label qui va permettre la vérification de la signature. Pour l'exemple, prenons la réponse suivante :

test.wildcard.jide. 60 IN A 131.254.200.7

.wildcard.jide. 60 RRSIG A 8 **2** 60 20120325141720 (

20120425141720 58034 wildcard.jide.

(signature ...)

)

La valeur 2 de ce champ, dans notre exemple, indique que la signature a été générée à partir des deux derniers labels du nom (wildcard.jide.). Grâce au champ Label, le résolveur déduit donc qu'il faut vérifier la signature en employant seulement les deux dernières étiquettes du nom (wildcard.jide.).

29. Voir : `https://en.wikipedia.org/wiki/Wildcard_DNS_record`.

Chaque enregistrement possède un Time To Live (TTL) associé. Lors de la signature d'un enregistrement, son TTL est pris en compte dans la signature. Lorsqu'un enregistrement est mis en cache par un résolveur ou envoyé dans une réponse, le TTL diminue. Il n'est donc plus possible de valider la signature à partir d'une seconde après sa réception car le TTL actuel et celui contenu dans la signature ne seront pas identiques. Le champ Original TTL permet de résoudre ce problème. Il contient la valeur du TTL de l'enregistrement couvert par la signature. Il permet ainsi de vérifier les signatures en remplaçant la valeur courante du TTL par la valeur originelle du TTL lors de la vérification.

La durée de validité de la signature est représentée par deux entiers codés 32 bits : l'un désigne la date de début de validité, l'autre désigne la date de fin de validité. Chaque entier représente le timestamp classique UNIX c'est-à-dire le nombre de secondes écoulées depuis le premier janvier 1970 à minuit. Dans le fichier de zone, la date de début et la date de fin apparaissent cependant au format AAAAMMJJHHMMSS. Le temps est en UTC et est donc indépendant des fuseaux horaires. Les secondes sont ignorées durant la vérification. Lorsqu'un résolveur validant reçoit une signature dont la période de validité est dépassée ou au contraire n'est pas encore atteinte, il doit la rejeter. On entrevoit ici une nouvelle difficulté : le temps des serveurs, aussi bien celui des serveurs récursifs que celui des serveurs faisant autorité doivent être synchronisés (NTP [30], c'est le bien). En effet, si les horloges des serveurs sont trop éloignés, des signatures peuvent être considérées, à tort, comme étant invalides. Les signeurs comme dnssec-signzone tentent de compenser les mauvaises pratiques par une date de début de validité fixée à une heure de moins que l'heure réelle de signature.

La signature est représentée en base64.

30. Protocole permettant la synchronisation d'une horloge via le réseau. Pour plus d'informations, voir : `https://fr.wikipedia.org/wiki/Network_Time_Protocol`.

3.2.3 Non-existence d'un enregistrement

Dans la version "classique" de DNS, le serveur faisant autorité retourne un code d'erreur comme « NXDOMAIN »/« NODATA » accompagné d'une absence de réponse (dans le sens où il n'y aura rien dans la section ANSWER) ainsi que le SOA de la zone pour le cache négatif lorsqu'on l'interroge pour un nom ou un type d'enregistrement qui n'existe pas.

Or, dans DNSSEC, pour pouvoir être authentifiée, la réponse doit être signée. La problématique est donc : comment signer le vide ?

DNSSEC introduit un nouvel enregistrement pour résoudre ce problème : NSEC (Next SECure). Celui-ci est mis à la fin de chaque nom (ex. : mycorporation.jide.) appartenant à la zone, peu importe si ce nom est présent dans la zone sous différents types (AAAA, NS, ...) et il indique, pour ce nom, le nom suivant dans l'ordre lexicographique. Le critère d'appartenance à la zone signifie que les enregistrements "glue" ne seront pas concernés par NSEC car il s'agit d'informations appartenant à une zone fille dupliquées dans la zone mère.

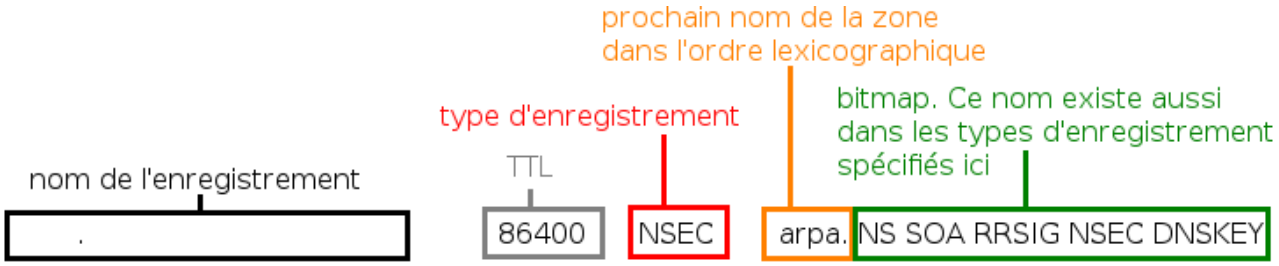

FIGURE 11 – Structure d'un enregistrement NSEC.

Durant la signature de la zone, les enregistrements sont classés par ordre alphabétique.

Si l'on interroge le serveur sur un nom qui n'existe pas, celui-ci va chercher l'encadrement dans lequel le nom se situe et renvoyer le NSEC correspondant à l'enregistrement qui sert de borne maximum à l'encadrement. Le résolveur regardera si la réponse correspond au nom demandé. Comme elle ne correspondra pas, il regardera si le nom demandé est dans l'intervalle [nom de l'enregistrement NSEC, prochain enregistrement indiqué dans le RDATA du NSEC]. Si c'est bien le cas, alors il a la preuve que le nom demandé n'existe pas.

Si le nom demandé existe mais que le type d'enregistrement n'existe pas, le NSEC correspondant au nom demandé sera retourné par le serveur faisant autorité. Le résolveur regardera alors le champ bitmap et verra que le type demandé n'est pas dans la liste et donc qu'il n'existe pas.

Le dernier enregistrement NSEC présent dans la zone pointe sur le premier enregistrement de la zone, c'est-à-dire l'enregistrement SOA. Le but est de créer une boucle.

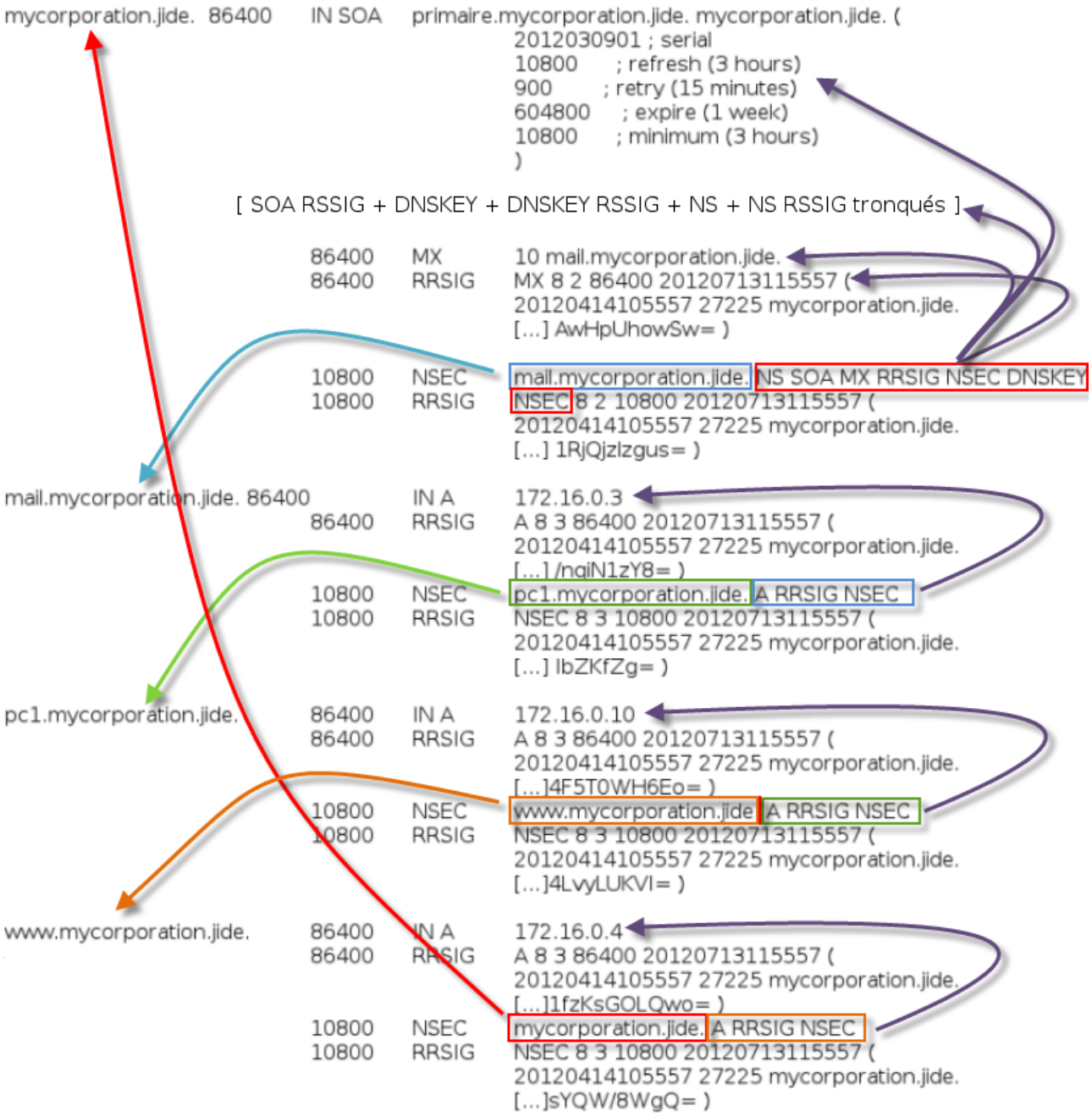

FIGURE 12 – Exemple de chaînage NSEC.

On se demande alors ce qui se passerait si l'on interrogeait le serveur sur l'enregistrement NSEC correspondant au SOA de la zone. On obtiendrait le prochain nom de la zone et la certitude qu'il n'y a pas de nom entre le SOA et le nom indiqué dans la réponse. Et si l'on interrogeait le serveur sur le NSEC du nom qu'il nous a retourné lors de notre première requête ? On obtiendrait le deuxième prochain nom de la zone. Et si l'on continuait comme cela jusqu'à temps de voir un enregistrement NSEC qui pointe sur le SOA ? On aurait ainsi parcouru toute la zone et obtenu tous ses enregistrements, sans même avoir recours à un transfert de zone. Cette pratique s'appelle le zone walking et peut-être mise en œuvre de manière très simple avec un utilitaire de test comme dig ou de manière automatisée avec des outils comme ldns-walk.

Cela peut poser de nombreux problèmes. On pense notamment aux zones dont le contenu n'est pas public et où ce dernier est vendu avec un contrat et des conditions d'utilisation (ex. : la zone fr. est rendue accessible par l'AFNIC pour 10 000 euros/an). On pense également aux enregistrements internes qui ne sont pas destinés à être publics (car ils dévoilent des services internes critiques de l'entité sur lesquels des attaques pourront être menées) même si les vues permettent également de se protèger contre ce type de nuisances depuis longtemps. On pense enfin à une entreprise qui réserve un nom de domaine pour le nouveau produit qu'elle prépare : un concurrent pourrait récupérer cette information sensible par un parcours de la zone du TLD et compromettre la commercialisation de ce nouveau produit (c'est aussi cela qui a amené l'AFNIC à empêcher le parcours de zone).

La solution consiste à utiliser un autre type d'enregistrement à la place de NSEC : NSEC3. Son but est toujours de signer la non-existence d'un nom mais sans permettre le parcours de zone. Son principe de fonctionnement est identique à celui de NSEC. On réalise toujours un chaînage mais cette fois-ci, on ne chaîne plus les enregistrements mais les condensats des enregistrements. Il n'est ainsi plus possible de parcourir la zone car il n'est pas possible de retrouver un texte clair à partir d'un condensat. Si un résolveur, pour une question portant sur un enregistrement A1, reçoit une réponse du type :
H0 NSEC3 H2 [...]
H0 RRSIG [...]
Il doit calculer H1, le condensat de A1 et vérifier que H0 < H1 < H2. Là encore, une notion d'ordre intervient et cette fois-ci, c'est les condensats qui sont ordonnés.

Cette technique ajoute deux types d'enregistrement : un enregistrement de type NSEC3-PARAM pour la zone et un enregistrement NSEC3 pour chaque nom appartenant à la zone. Le critère d'appartenance à la zone signifie que les enregistrements "glue" ne seront pas concernés par NSEC3.

NSEC3PARAM s'applique à la zone et contient la fonction de hachage et les paramètres à utiliser. Il permet au(x) serveur(s) faisant autorité sur la zone de calculer les condensats des noms ainsi qu'à déterminer les enregistrements NSEC3 à renvoyer lors d'une requête sur un nom qui n'existe pas. Un enregistrement NSEC3PARAM n'est donc pas utilisé par les serveurs résolveurs.

Un champ supplémentaire, indiquant la longueur du sel, existe mais il n'est pas représenté dans le fichier de zone.

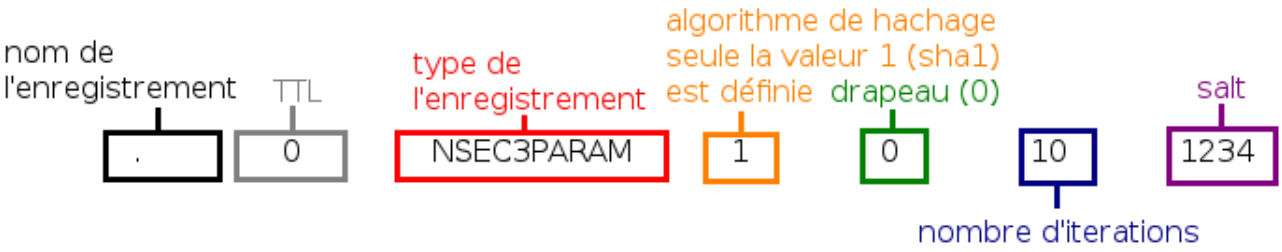

FIGURE 13 – Structure d'un enregistrement NSEC3PARAM.

Le nombre d'itérations définit le nombre de fois où la fonction de hachage sera appliquée. Tout comme le sel, cela permet d'améliorer la résistance des condensats à une attaque par dictionnaire/rainbow-table [31] (une personne malveillante pourrait récupérer la liste des NSEC3 d'une zone, générer, hors-ligne, l'ensemble des condensats possibles pour une zone et les vérifier dans un deuxième temps). En contrepartie, NSEC3 augmente le temps de validation par les résolveurs. C'est pourquoi un nombre maximum d'itérations est fixé, par le RFC 5155, en fonction de la taille de la plus petite clé utilisée dans la zone (ex. : si la plus petite clé de la zone est une RSA1024, alors le nombre d'itérations des condensats ne doit pas dépasser 150).

Le sel doit être changé régulièrement, plus rapidement que le temps nécessaire à un éventuel attaquant pour générer un dictionnaire ou une rainbow-table ayant un taux de couverture idéal pour la zone considérée. Typiquement, le RFC 5155 recommande de changer le sel à chaque signature de la zone.

NSEC3 s'applique à chaque nom appartenant à la zone. Ce critère d'appartenance à la zone signifie que les enregistrements "glue" ne seront pas concernés par NSEC3.

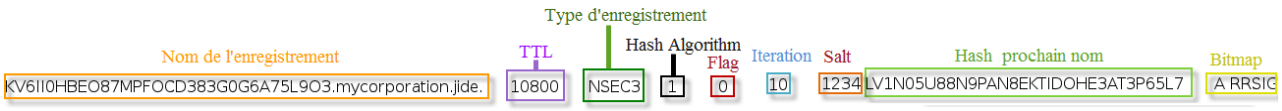

FIGURE 14 – Structure d'un enregistrement NSEC3.

Aucun champ n'est nouveau par rapport à NSEC et NSEC3PARAM.

Cependant, le champ « drapeaux » peut prendre une autre valeur que 0 alors que cette valeur est fixe dans l'enregistrement NSEC3PARAM. En effet, il peut prendre la valeur 1 pour indiquer l'utilisation de l'option opt-out. Elle permet aux enregistrements NSEC3 de "sauter" par dessus, c'est-à-dire d'ignorer, les délégations non signées. Dans les zones contenant beaucoup de sous-zones non signées (comme un TLD), cette option permet de signer beaucoup plus rapidement la zone (un facteur de 30 peut être constaté en fonction de la zone [32]) et de diminuer la taille

31. Pour une définition simplifiée, voir : `https://fr.wikipedia.org/wiki/Table_arc-en-ciel`.

du fichier de zone signé. Néanmoins, cela se fait au détriment de la sécurité puisque dans ce cas, il ne sera plus possible de prouver la non-existence d'une sous-zone non signée.

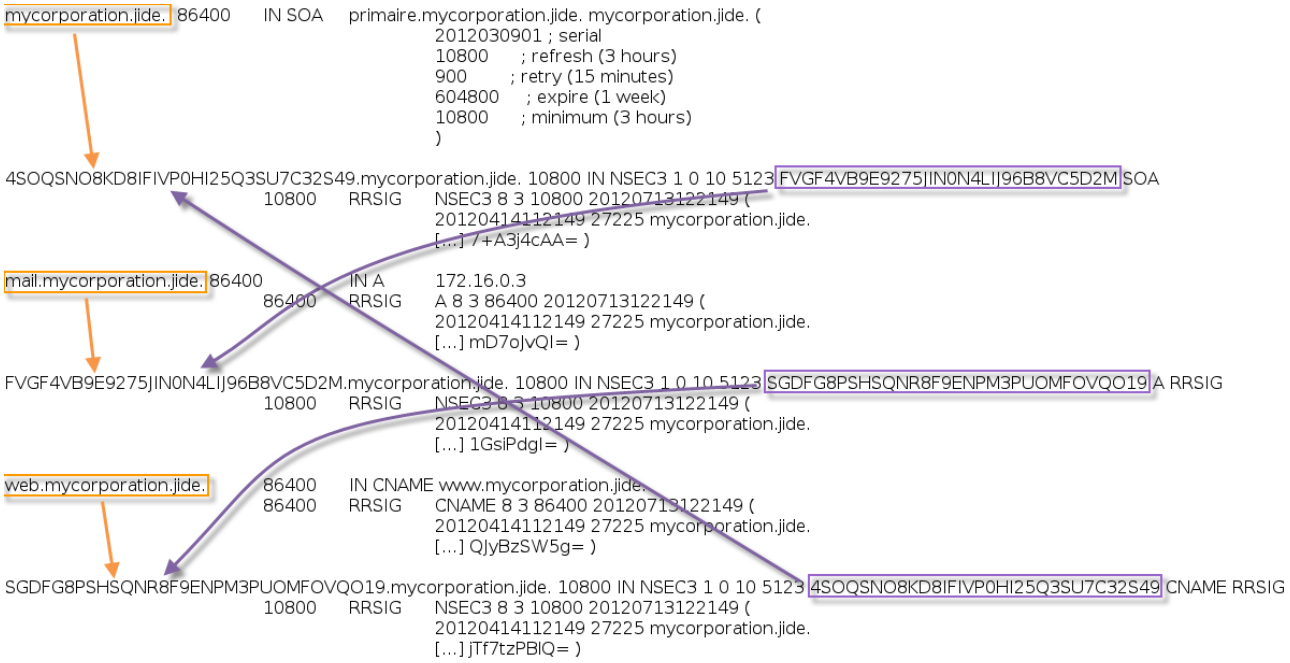

FIGURE 15 – Exemple de chaînage NSEC3.

32. Source : `http://www.bortzmeyer.org/5155.html`.

3.2.4 Délégation et chaîne de confiance

Nous avons vu que le résolveur doit posséder les parties publiques des clés ayant servi à signer la zone qu'il veut valider. La problématique est donc de savoir comment il va avoir avoir connaisance des clés publiques des zones.

Dans un premier temps, nous pouvons faire deux propositions :

- Diffuser les clés de toutes les zones à tout le monde via un autre canal que le DNS. Cette solution ne résiste évidemment pas au passage à l'échelle. En effet, il est impossible de distribuer les clés à tous les utilisateurs, même en utilisant le même mécanisme que pour la diffusion des adresses IPs des serveurs de la racine car, d'une part, les clés changent et, d'autre part, la moindre erreur de saisie/recopie/copier-coller entraîne des erreurs de validation. De plus, on imagine mal les administrateurs de serveurs récursifs faire la liste des zones signées et aller récupérer les clés et ce, de manière régulière.
- On peut diffuser les clés publiques dans le DNS. C'est d'ailleurs à cela que servent les enregistrements de type DNSKEY. Mais on obtient alors un problème d'oeuf et de poule : puisque le DNS n'est pas sûr, on ne peut pas avoir confiance dans les clés distribuées.

La solution est d'utiliser la deuxième proposition mais en y ajoutant une subtilité : on va avoir recours au principe arborescent du DNS et créer une chaîne de confiance. On va partir d'un point précis et déléguer la confiance à partir de ce point. Exemple : la racine, le TLD fr et la sous-zone afnic.fr sont signées. La racine va déléguer la confiance à fr. et fr. va déléguer la confiance à afnic.fr. [33]. Ainsi, nous avons besoin uniquement de la partie publique de la KSK de la racine qui sert ici de point d'entrée. Dans la littérature DNSSEC, ce type de point d'entrée est appelé un trust anchor, ancre de confiance en français. On comprend donc que, depuis sa signature, la racine de l'ICANN est devenue l'ancre de confiance principale du DNS mais pas la seule, nous y reviendrons.

La problématique devient donc : comment déléguer la confiance entre les zones ayant un lien de parenté ? DNSSEC introduit un nouvel enregistrement, DS (Delegation Signer). Cet enregistrement joue le rôle de maillon de confiance entre une zone parente et une zone fille.

33. Nous ne tiendrons pas compte du fait que fr. et afnic.fr. sont gérées par le même organisme, à savoir l'AFNIC. Cela ne nuit pas à la démonstration. Si plus de zones étaient signées, nous en choisirions une autre avec plaisir.

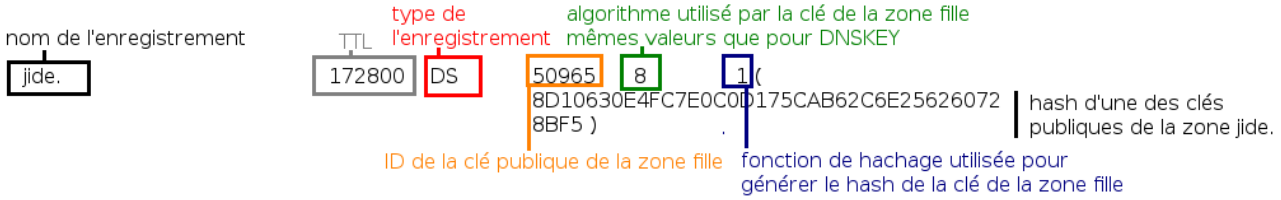

FIGURE 16 – Structure d'un enregistrement DS.

Il stocke, dans une zone parente, le condensat de la partie publique de la KSK actuellement utilisée (ou amenée à être utilisée) par la zone fille. Le DS d'une zone [34] est créé par le logiciel signeur durant la signature de la zone. Il convient de les transférer au gestionnaire de la zone parente via un canal sécurisé (afin d'éviter tout risque d'altération durant le transport).

On comprend donc qu'en fin de chaîne, et lorsque le déploiement de DNSSEC sera total, le résolveur devra avoir connaissance de seulement une clé publique : la KSK de la racine. Les autres clés seront récupérées et validées par le résolveur à mesure qu'il avance la résolution du nom de domaine demandé.

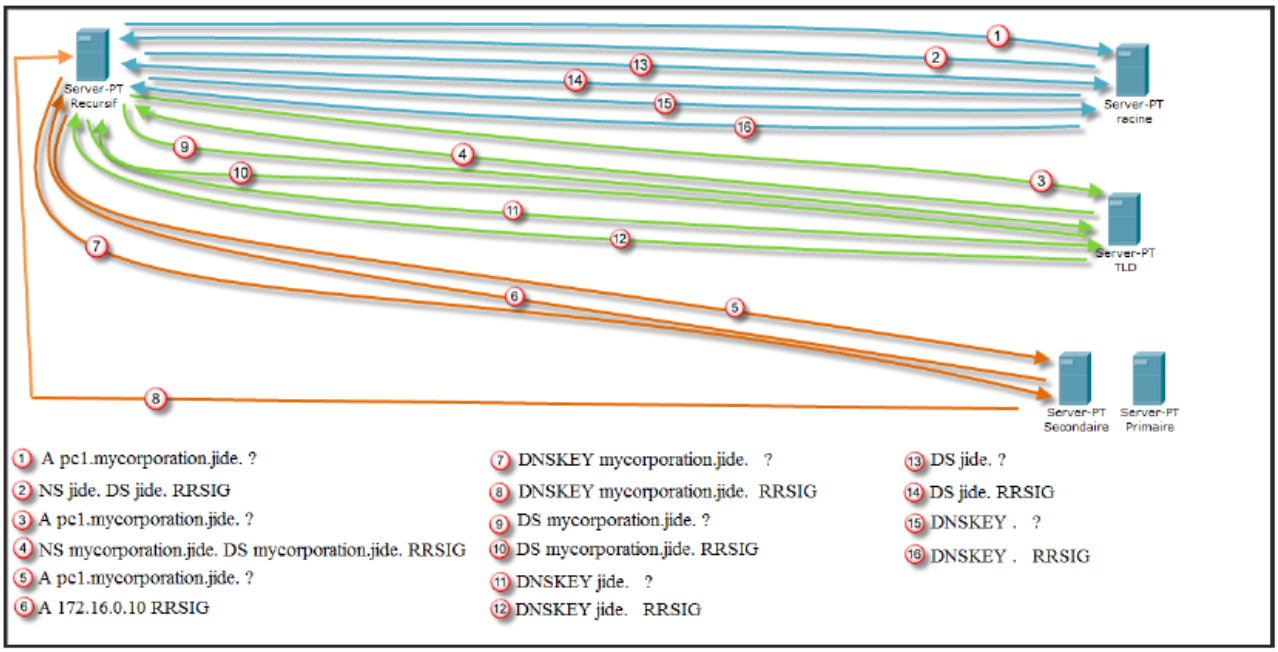

FIGURE 17 – Exemple d'une résolution de nom avec DNSSEC.

34. Pour ceux qui se demandent pourquoi dnssec-signzone génère deux enregistrements DS par zone, sachez que les deux enregistrements concernent bien la même clé : la KSK de la zone fille en activité au moment de la signature. La seule différence entre les deux enregistrements est que l'un des condensats utilise SHA1, l'autre utilise SHA256. SHA256 est utilisé pour des raisons de sécurité (SHA1 étant considéré comme dangereux depuis 2005). Le DS SHA1 est conservé pour des raisons de compatibilité avec tous les résolveurs mais il n'est pas indispensable.

En attendant la signature complète de l'arborescence, on se retrouve avec des îlots sécurisés. Comment diffuser les ancres de confiance de ces îlots ? Comment les quelques organismes qui ont voulu tester le déploiement de DNSSEC avant la signature de la racine ont-ils procédé ? Et si je veux signer ma zone mais que ma zone parente n'est pas signée ? Et si mon registar (ou mon registry) n'accepte pas la soumission d'enregistrements de type DS ?

On peut envisager de distribuer les clés mais on l'a vu ci-dessus, cette méthode ne résiste pas au passage à l'échelle. Une solution existe : DLV (DNSSEC Lookaside Validation). L'idée est de séparer l'arborescence classique contenant les enregistrements de l'arborescence contenant les délégations signées.

Plusieurs registres DLV existent mais le plus important est sans nul doute celui de l'ISC (dlv.isc.org). Plusieurs racines peuvent signer un même domaine. On peut avoir, par exemple, une racine DLV qui signe .org et une autre qui signe example.org. Plusieurs algorithmes de choix de la meilleure racine à utiliser en cas de recouvrement sont décrits dans le RFC 5074, le plus simple étant décrit comme "choisir la racine la plus spécifique" (celle de example.org dans notre exemple).

L'inconvénient de cette solution est qu'elle nécessite une modification de la configuration des deux côtés : le gestionnaire de la zone doit avoir déposé les enregistrements de sa zone dans un répertoire DLV et le résolveur doit être configuré pour utiliser le bon répertoire DLV c'est-à-dire celui dans lequel la clé de la zone désirée se trouve.

DLV reprend les enregistrements DS et en change juste le nom : DLV au lieu de DS. Le formalisme de l'enregistrement est strictement identique. Le principe de fonctionnement et la structure de l'enregistrement restent identiques. Tout comme les enregistrements DS, les DLV seront générés par le logiciel signeur et seront à transmettre, de manière sécurisée, au registre DLV que l'on souhaite utiliser.

Lors de la résolution, le résolveur demande l'enregistrement DS d'une zone à la zone parente. Si aucune réponse satisfaisante ne lui parvient et s'il est configuré pour, il va interroger un registre DLV. Pour la zone msi.unilim.fr., en utilisant le registre DLV de l'ISC, les enregistrements DLV correspondant à la zone se trouveront en msi.unilim.fr.dlv.isc.org.

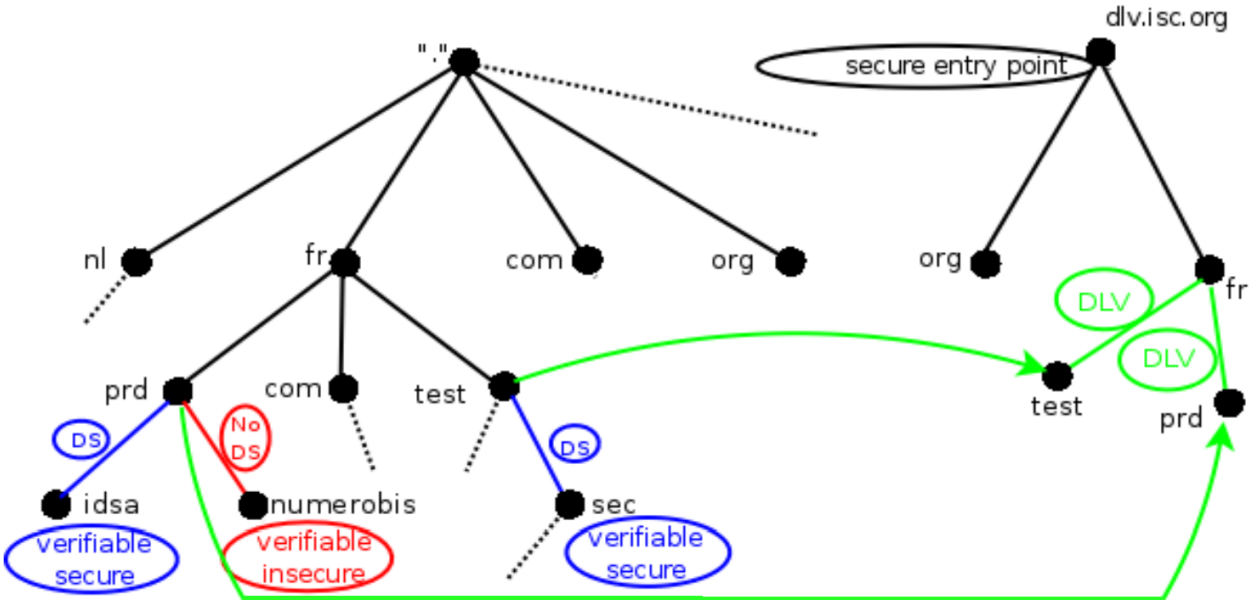

FIGURE 18 – Schéma de fonctionnement de DLV. D'après la présentation de Stéphane Bortzmeyer aux JRES 2009.

Une dernière interrogation peut se poser : et si je ne veux pas utiliser DLV mais juste faire un échange de clés avec une ou deux zones ? Et de toute façon, il faut bien mettre à jour la clé publique de la racine dans les résolveurs et autres clients, non ? Comment faire ?

En effet, il faut tenir à jour les clés publiques de la racine sous peine de ne plus pouvoir valider aucun domaine lorsque les clés changeront. Un mécanisme a été prévu pour cela dans le RFC 5011 : la nouvelle KSK d'une zone sera signée avec l'ancienne KSK avant sa prépublication dans le fichier de zone. Ainsi, tous les clients accepteront cette nouvelle clé. Les clients se construisent une chaîne de confiance en partant de la clé avec laquelle ils ont été configurés jusqu'à la dernière clé publiée.

Ce mécanisme induit des risques. Le plus important étant lors d'un vol d'une clé privée : l'attaquant pourra continuer de créer une chaîne de confiance valide et le retrait de la paire de clés compromise de la zone ne suffira pas. Ce scénario peut être rendu plus dramatique si l'on considère que le gestionnaire de la zone ne se rend pas immédiatement compte du vol de la clé privée. Cependant, le RFC 5011 introduit deux mécanismes pour limiter cela :

– Un bit du champ « drapeaux » de l'enregistrement DNSKEY permet d'indiquer que la KSK concernée par l'enregistrement a été révoquée et que les clients doivent donc l'enlever de leur chaîne de confiance et ne plus l'utiliser. Attention : ce bit permet la révocation uniquement dans le cas d'une clé utilisée comme première trust-anchor par un résolveur. Autrement dit : il ne sert à rien dans le cadre d'une délégation avec un enregistrement

DS. Ce bit permet d'avoir deux valeurs supplémentaires pour le champ « drapeaux » : 384 (ZSK révoquée) et 385 (KSK révoquée).

– Un délai d'attente (de typiquement 30 jours) avant l'acceptation de la clé par un résolveur, permet d'éviter que l'attaquant ne fasse accepter une nouvelle clé immédiatement. Durant ce temps d'attente, le client doit effectuer une vérification périodique de la clé (dont la méthodologie complète est détaillée dans le RFC). BIND implémente cette fonctionnalité sous le terme de « managed-keys » par opposition aux « trusted-keys » qui sont statiques et non-mises à jour par le logiciel.

3.2.5 Limites

Le premier frein à une utilisation massive de DNSSEC dans les prochains mois est que la très grande majorité des zones ne sont pas signées à l'heure actuelle et ne le seront pas avant longtemps.

Cela est dû à la complexité de DNSSEC et notamment à la gestion des clés. Le débogage des erreurs est également fortement complexifié. Il en va de même pour les résolveurs : deux lignes à ajouter dans le fichier named.conf suffisent à activer la validation DNSSEC mais si des erreurs se produisent en amont, le résolveur ne validera plus les signatures et les utilisateurs du résolveur taperont sur les responsables du résolveur.

Certains organismes y ajoutent également une notion de coût humain supplémentaire : s'occuper d'un serveur DNS génère un coût fixe : on configure le serveur une bonne fois pour toutes. S'occuper d'un serveur utilisant DNSSEC génère, en plus du coût fixe d'installation, un coût récurrent : il faut maintenir le serveur, vérifier le bon roulement des clés, résoudre les problèmes, ...

Le deuxième frein sera dû aux pare-feux et autres intermédiaires réseaux mal-conçus et/ou mal configurés qui bloquent les paquets DNS d'une taille supérieure à 512 octets alors que cette limite a pourtant été rendue obsolète par le RFC 2671 publié en 1999 et/ou qui bloquent l'usage de DNS sur TCP. Les réponses sécurisées dépassent facilement cette taille fatidique de 512 octets. Dans un tel cas, le bit TC (truncated), qui sert à indiquer au client qu'un RRset faisant partie de la réponse (section ANSWER uniquement) n'a pas pu être intégré totalement dans la réponse, n'est d'aucune utilité : si le résolveur n'a pas indiqué qu'il ne pouvait pas recevoir plus de 512 octets, le paquet, jugé trop grand par le pare-feu, sera supprimé par ce dernier. Si le résolveur indique qu'il ne peut pas recevoir plus de 512 octets et qu'il reçoit le bit TC mais que le pare-feu bloque DNS sur TCP, le résolveur ne pourra pas non plus procéder à la résolution.

Mais le problème ne s'arrête pas là. En effet, les réponses sécurisées peuvent même dépasser 1500 octets. À titre d'exemple, une simple requête sur les serveurs racines génère une réponse de plus de 1100 octets. Or, la MTU classique sur internet est justement de 1500 octets. Au-delà de cette limite, il est nécessaire de fragmenter la réponse en plusieurs paquets IP. Or, certains pare-feux ou intermédiaires (middlebox) ne gèrent pas les fragments et/ou bloquent totalement l'ICMP (qui sert, dans le cas présent, à avertir que le paquet doit être fragmenté via son type 3, code 4). Le bit TC n'est, là encore, d'aucune aide. En effet, avec EDNS0, on peut, en temps normal, avoir une réponse de 4k au maximum. Au niveau DNS, je peux très

bien avoir une réponse supérieure 1500 octets et inférieure à 4k. Le flag TC ne sera donc pas positionné mais la réponse sera fragmentée au niveau IP. Si les fragments sont ignorés sur le chemin ou si ICMP est ignoré à un endroit du chemin alors que le paquet doit être fragmenté, le client ne recevra pas la réponse dans son intégralité voire par du tout. On peut supposer des mécanismes de contrôle côté client comme « si les compteurs en début de réponse ne sont pas égaux au nombre d'enregistrements réellement reçus, alors il y a eu un problème et on rejoue la requête en TCP » (en supposant bien sûr qu'un pare-feu ne bloque pas le protocole DNS sur TCP) sans savoir si de tels mécanismes ont été mis en place.

Les réseaux équipés de tels pare-feux/intermédiaires rencontreront des problèmes lors de l'usage de DNSSEC et seront donc un frein à son utilisation massive.

Enfin, une limite de DNSSEC peut apparaître en fonction de l'endroit où est effectuée la validation des signatures cryptographiques. En effet, dans tout ce rapport et dans notre mise en œuvre, nous avons supposé que le meilleur endroit pour effectuer cette validation est le serveur résolveur. Mais, les réseaux situés entre le résolveur et le client final ne sont pas sûrs et des attaques peuvent très bien s'y produire.

De plus, on sait très bien que les résolveurs, et notamment ceux des FAI, sont les premiers à falsifier les réponses que ce soit à des fins publicitaires (DNS menteur [35]) ou à des fins de censures (cas de la LOPPSI 2 [36] ou de l'affaire copwatch-idf [37] en France, par exemple).

Enfin, contrairement à l'idée reçue, le bit AD (Authentic Data, qui indique que le résolveur a validé les signatures) ne permet pas de garantir que les données sont sûres ni qu'elles n'ont pas été altérées entre le résolveur et le client final car ce bit n'est pas protégé par DNSSEC et peut donc être manipulé.

Deux solutions sont possibles : sécuriser le canal entre le résolveur et le client (avec des technologies comme IPSEC ou, plus spécifique au DNS, SIG/TSIG) ou effectuer la validation DNSSEC en bout de chaîne, sur le client final.

La première solution ne résout pas le problème de la masse : le FAI de M. ToutLeMonde ne sécurisera pas le canal et rien ne prouvera non plus la fiabilité du résolveur mis en place par FAI.

La deuxième solution amène plusieurs questions.

35. Pour une définition, voir : `http://www.bortzmeyer.org/dns-menteur.html`.

36. Pour une présentation de l'article 4 de cette loi, voir : `http://www.pcinpact.com/news/62402-conseil-constitutionnel-loppsi-article-filtrage.htm`.

37. Copwatch-idf, site web qui se voulait être un révélateur de la violence policière en France, utilisait des méthodes discutables (publications de données personnelles, entre autres). Il s'est retrouvé temporairement censuré en France. Voir : `http://www.pcinpact.com/news/66468-copwatch-blocage-fai-dns-ip.htm`.

D'une part, tous les clients ne peuvent pas procéder à une validation cryptographique, on pense notamment aux mobiles et au monde de l'embarqué, même si cette situation va évoluer avec l'augmentation constante de la puissance de ces appareils.

D'autre part, cela augmenterait fortement la charge sur les serveurs faisant autorité puisqu'il n'y aurait plus de caches communs partagés entre un ensemble d'utilisateurs. Cet effet est à relativiser puisque les serveurs faisant autorité fréquemment interrogés (ceux de la racine, ceux de l'AFNIC, ...) sont déjà surdimensionnés pour répondre à d'éventuelles attaques par déni de service.

De plus, l'offre logicielle permettant une mise en place facile d'une validation DNSSEC à la maison par M. Toutlemonde est presque inexistante. Signalons toutefois l'existence du logiciel dnssec-trigger [38] qui se veut accessible au plus grand nombre, qui est multi-systèmes, qui évite de monter un serveur résolveur local quand c'est possible (c'est-à-dire quand le résolveur du réseau est de bonne qualité) et qui valide quand même les signatures chez le client final. Une autre alternative intéressante et transparente pour les particuliers serait que les FAI activent la validation sur les forwarders DNS qui se trouvent dans les box qu'ils louent. Cette alternative est, malheureusement, utopique à la vue des pratiques passées et actuelles.

Enfin, rien n'empêche les programmeurs d'inclurent dans leurs logiciels un résolveur/validateur grâce à des bibliothèques telles que libunbound. Il faut alors se demander s'il convient de faire cette validation dans chaque logiciel (en terme de coût). Il faut également prendre en considération les parcs informatiques rarement mis à jour.

Pour résumer, tant que la validation ne sera pas effectuée en fin de chaîne ou que le canal final ne sera pas sécurisé, un risque de corruption des données est possible et la sécurité apportée par DNSSEC en est réduite d'autant.

38. Pour une présentation de dnssec-trigger, voir : `http://www.bortzmeyer.org/dnssec-trigger.html`.

3.3 Mise en oeuvre

Nous reprenons ici notre maquette DNS réalisée sous Netkit et nous mettons en place DNSSEC. Ainsi nos réseaux, nos machines et notre hiérarchie des noms présentées dans la section « Mise en oeuvre » de la partie « DNS » restent les mêmes.

Le but est d'obtenir une hiérarchie totalement signée, de la racine jusqu'à la zone terminale mycorporation.jide. Le seul point d'entrée sécurisé sera donc la racine. Nous n'utilisons pas DLV et nous ne mettrons pas en place un roulement des clés.

Plusieurs outils existent pour mettre en place DNSSEC et notamment pour signer les zones mais nous avons utilisé les utilitaires fournis de base avec BIND car ils sont amplement suffisants pour notre installation.

3.3.1 Génération des clés :

Dans un premier temps, nous devons générer les paires de clés pour chacune de nos zones. Ci-dessous, la paire de commandes qui génère une paire de clés :

- dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE -r /dev/random <nom-de-la-zone>
- dnssec-keygen -3 -a RSASHA256 -b 1024 -n ZONE -r /dev/random <nom-de-la-zone>

-f : Précise que la clé sera une KSK et son absence signifie que la clé sera une ZSK.

-3 : Génère une ZSK qui sera compatible avec l'utilisation de NSEC3. Si l'on utilise pas NSEC3 dans la zone, alors cette option est inutile.

-a : Permet de choisir les algorithmes cryptographiques utilisés. Selon les documents officiels, la racine de l'ICANN utilise l'algorithme RSASHA256 pour les KSK et les ZSK. L'usage de RSASHA256 est également recommandé pour toutes les zones. Nous avons suivi cette recommandation.

-b : Permet de spécifier la taille des clés. Les documents officiels indiquent que les KSK de la racine ICANN ont une longueur de 2048 bits alors que les ZSK de la racine ont une longueur de 1024 bits. Il est recommandé d'adapter la taille des clés en fonction de l'importance de la zone sans aller en dessous de 1024 bits. Ces recommandations datant d'un peu plus de cinq ans maintenant et prenant en compte le fait que le document lui-même ne garantit pas la suffisance des clés d'une taille de 1024 bits qu'il préconise au-delà des cinq ans à partir de son écriture, nous avons choisi de prendre des clés de 2048 bits pour la KSK et 1024 bits pour la ZSK et ce, pour toutes nos zones.

-n : Indique le type de ce qui sera signé (ZONE pour une zone, HOST pour une machine (ce dernier est utilisé uniquement pour les transferts de zone)).

-r : Permet de choisir le générateur de nombres aléatoires qui sera utilisé. Ce paramètre est facultatif car, par défaut, dnssec-signzone utilise /dev/random. La génération de clés demande beaucoup d'entropie. Vous pouvez utiliser un générateur d'entropie matériel [39] ou logiciel [40] pour accélérer la création des clés et des signatures. Vous pouvez également utiliser /dev/urandom mais uniquement si votre but est de créer une maquette car /dev/urandom est déconseillé pour toutes les opérations cryptographiques.

Nous avons donc utilisé les commandes suivantes pour générer toutes les paires de clés de notre maquette :

mycorporation.jide. :

- dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE mycorporation.jide
- dnssec-keygen -3 -a RSASHA256 -b 1024 -n ZONE mycorporation.jide

16.172.in-addr.arpa. (reverse de mycorporation.jide.) :

- dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE 16.172.in-addr.arpa
- dnssec-keygen -3 -a RSASHA256 -b 1024 -n ZONE 16.172.in-addr.arpa

jide. :

- dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE jide
- dnssec-keygen -3 -a RSASHA256 -b 1024 -n ZONE jide

in-addr.arpa. :

- dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE in-addr.arpa
- dnssec-keygen -3 -a RSASHA256 -b 1024 -n ZONE in-addr.arpa.

. :

- dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE .
- dnssec-keygen -a RSASHA256 -b 1024 -n ZONE .

39. Exemple : Entropy Key. Voir : `http://jpmens.net/2012/01/24/entropy-random-data-for-dnssec/`.
40. Exemple : Haveged. Voir `http://www.issihosts.com/haveged/`.

arpa. :

- dnssec-keygen -f KSK -a RSASHA256 -b 2048 -n ZONE arpa
- dnssec-keygen -a RSASHA256 -b 1024 -n ZONE arpa

3.3.2 Signature des zones

La commande dnssec-keygen retourne deux fichiers : l'un, nommé « *.private » contient la clé privée qu'il faut garder ... privée. L'autre contient la clé publique et plus précisément l'enregistrement DNSKEY. Il faudra donc incorporer les deux enregistrements DNSKEY (celui de la ZSK et celui de la KSK) dans le fichier de zone sans oublier d'incrémenter le serial du SOA de la zone.

Ensuite, il faut signer la zone avec la commande suivante : dnssec-signzone -o <nom-de-la-zone> -e now+7776000 -3 <salt> <fichier-de-la-zone>

-e : Permet de spécifier la durée de validité des signatures. (ici : trois mois). Dans la racine officielle de l'ICANN, les signatures des enregistrements de type DNSKEY ont une durée de vie de quinze jours alors que les signatures des enregistrements ayant un autre type (A, AAAA, NS, ...) ont une durée de vie de sept jours. Pour Stéphane Bortzmeyer (ingénieur à l'AFNIC) que nous avons interrogé à ce sujet, cette différence n'a aucune justification technique et est probablement d'ordre politique (la KSK de la racine étant gérée par l'ICANN alors que la ZSK est gérée par Verisign). Pour les autres zones, on recommande une durée variable en fonction de la zone et de son TTL tout en conseillant une durée de vie d'une semaine. Nous avons arbitrairement choisi une durée de vie de trois mois afin de ne pas passer notre temps à resigner nos zones.

-3 : Permet de spécifier le sel qui sera utilisé pour la génération des condensats NSEC3. Il faut l'écrire en hexadécimal. Une taille de 8 caractères est recommandée. Si l'on n'utilise pas NSEC3, on n'a pas besoin de cette option.

Cette commande retourne deux fichiers : un fichier de la forme <nom de la zone>.signed, qui contient le fichier de zone signé et un fichier dsset contenant les enregistrements DS à mettre dans le fichier de zone de la zone parente avant de la signer.

Voici donc les commandes que nous avons utilisées pour signer nos zones :

mycorporation.jide. :

- dnssec-signzone -o mycorporation.jide -e now+7776000 -3 27206418 master.mycorporation.jide

16.172.in-addr.arpa. :

- dnssec-signzone -o 16.172.in-addr.arpa -e now+7776000 -3 35277615 master.172.16.rev

jide. :

- dnssec-signzone -o jide -e now+7776000 -3 17188056 master.jide

in-addr.arpa. :

- dnssec-signzone -o in-addr.arpa -e now+7776000 -3 73078940 master.in-addr.arpa

. :

- dnssec-signzone -o . -e now+7776000 master.racine

arpa. :

- dnssec-signzone -o arpa -e now+7776000 master.arpa

3.3.3 Modification des fichiers named.conf

Sur les serveurs faisant autorité :

1. Il faut ajouter l'option « dnssec-enable yes ; » dans la section « options ».
2. Il faut également indiquer que la zone est désormais dans le fichier .signed et que c'est celui-ci qu'il faut utiliser pour servir la zone.
3. Il faut redémarrer le démon named ou lui faire recharger le fichier de configuration pour prendre en compte ces modifications.

Sur le serveur récursif :

1. Il faut ajouter l'option « dnssec-enable yes ; » dans la section « options ».
2. Il faut rajouter une section « trusted-keys » qui contient la KSK de la racine :
   trusted-keys "." 257 3 8 "AwEAAbgsKZQiGGTPSviuRVN
   uZvdfeEIu9+69h9uMVJpF9SsNd7lAiGwK
   YvyNkC3I4bSmAOzFfU3PN5JCcPTysL12cJIFFPUnzVml3J4gssJjcMgv
   bBHg+iJr+SG++B80Wp4j
   5yB+mD+QGyfsNjVusWpgLiL7dwo4nr42YKP2
   TTNM9K7ZtMJgLEkO7byMDtnKowqCM5mPAB6
   el9zqn5jARNZnz0v/G4+3
   tJScBpUWtplrPrMZWmMUapBAaNeQDkLSG/f58yXPwGwZQfMhIkOZhAgc
   vw58hQmN+MlHDLilbu5nT6RY9eCPTpzkh07ut22LH1uS1lI6hwkgiEy4
   kRgNTW1UCl0=" ; ;

   Note : Une section « managed-keys » est une implémentation du RFC 5011 disponible depuis la version 9.7. Elle autorise BIND à effectuer lui-même le suivi des clés en cas de roulement. Dans ce cas, on ne fournit que la clé initiale. Comme nous ne prévoyons pas de changer nos clés, nous n'avons pas besoin que BIND fasse un suivi des clés et nous n'avons donc pas besoin d'une section « managed-keys ».
3. Il faut activer la validation des signatures en rajoutant un paramètre dans le bloc des options : dnssec-validation yes ;.

   Note : Dans le cas d'une section « managed-keys », il faut utiliser une directive « dnssec-validation auto » à la place.
4. Il faut redémarrer le démon named ou lui faire recharger le fichier de configuration pour prendre en compte ces modifications.

Sur la box : Il suffit de rajouter l'option « dnssec-enable yes ; » dans la section « options » du fichier named.conf et de recharger le fichier de configuration.

3.3.4 Validation de notre maquette

Pour valider notre maquette DNSSEC, nous avons de nouveau utilisé le logiciel Zonecheck. En effet, ce logiciel dispose d'une fonctionnalité de contrôle de la bonne mise en place de DNSSEC depuis sa version 3. Zonecheck ne nous a remonté aucune erreur concernant nos zones.

Nous avons également validé manuellement la configuration de notre racine et de notre TLD en comparant, avec dig, les résultats obtenus sur la racine officielle de l'ICANN et sur quelques TLDs (fr., net., info.). Nous n'avons remarqué aucune anomalie.

4 Faiblesses

4.1 Motivations

Comme la quasi-totalité de toutes les transactions sur internet commence par une résolution DNS, il est très tentant d'attaquer ce système puisque les possibilités sont considérables.

On peut ainsi imaginer un ensemble d'attaques ayant un degré de malveillance variable allant d'un jeune qui s'amuse ; à un détournement du trafic d'un site web (ex. : celui d'une banque) vers un site falsifié ; en passant par une entreprise qui empêche l'accès au site web de son concurrent direct.

Pour résumer : on attaque DNS for fun and profit ! (pour le fun et le profit)

4.2 Généralités

En dehors des failles d'implémentation (nous y reviendrons), les faiblesses du protocole DNS se résument à l'image ci-dessous :

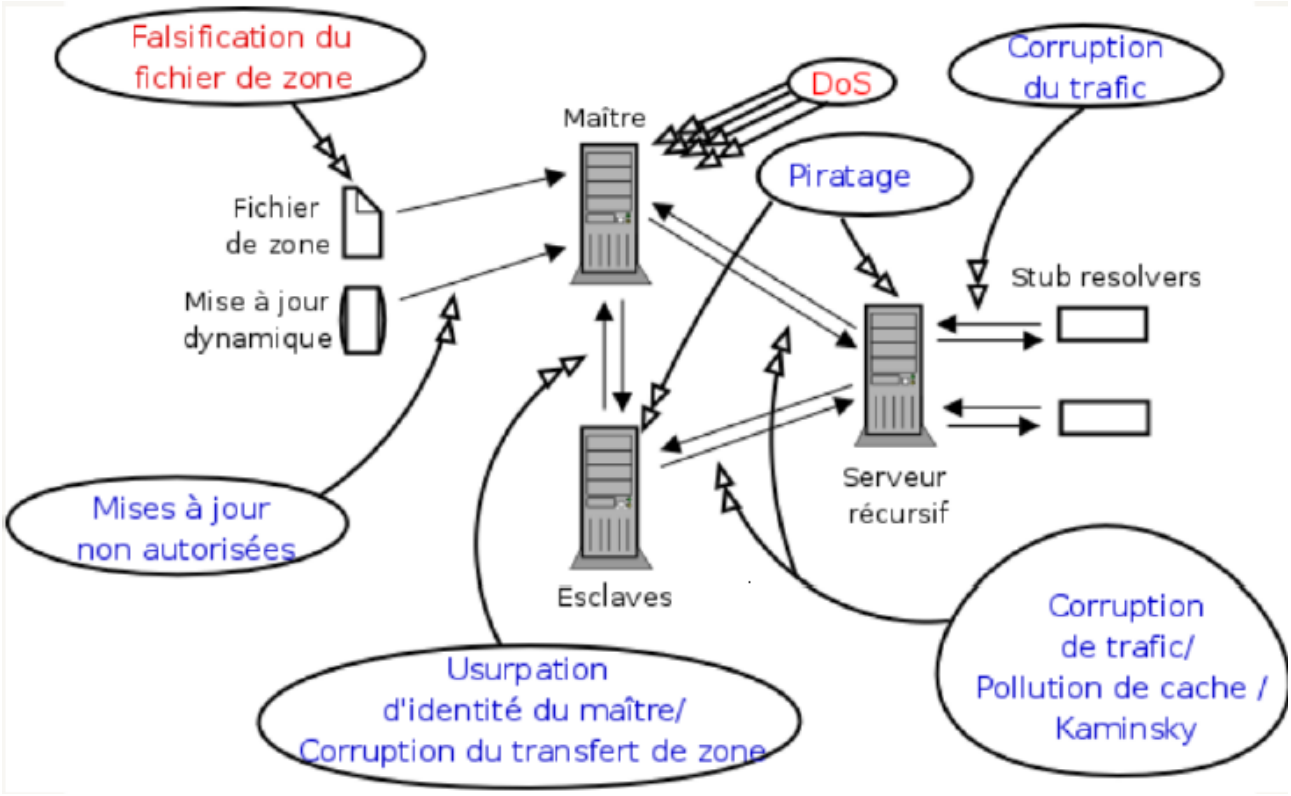

FIGURE 19 – Faiblesses du DNS. Source : présentation de Stéphane Bortzmeyer à SARSSI 2009.

D'une part, les attaques marquées en rouge sur ce schéma ne peuvent être corrigées ni dans le protocole standard, ni par DNSSEC.

En effet, si un registry ou un registar se fait pirater ou commet une erreur (par exemple, il y a quelques années, le domaine sex.com avait obtenu une autorisation de transfert suite à un fax falsifié), le fichier de zone contiendra une/des erreur(s). DNSSEC ne détecte pas de telles erreurs (ce serait d'ailleurs impossible pour la plupart d'entre elles) et signe la zone. Une zone signée ne garantit pas que les informations qu'elle contient sont correctes mais uniquement qu'elles arriveront sans altération jusqu'au résolveur validant.

De plus, une attaque par déni de service (DoS) n'est pas spécifique au protocole DNS et ne peut donc pas être détectée ni corrigée par DNSSEC. D'autres solutions sont à déployer (pare-feu, anycast, NIDS [41], ...) pour atteindre cet objectif.

Par ailleurs, nous rappelons qu'un man-in-the-middle (attaque de l'homme du milieu, en français) [42] est empêchée par DNSSEC à condition que la validation des enregistrements soit

41. Pour une définition, voir : `https://en.wikipedia.org/wiki/Network_intrusion_detection_system`.

effectuée sur le client ou qu'une méthode de sécurisation du canal (IPSec par exemple) soit déployée entre la machine de validation et le client final. Si cette validation est déportée, il y a un risque uniquement entre le serveur de déport et le client final.

D'autre part, DNSSEC peut corriger trois autres types d'attaques (barrées sur le schéma) :

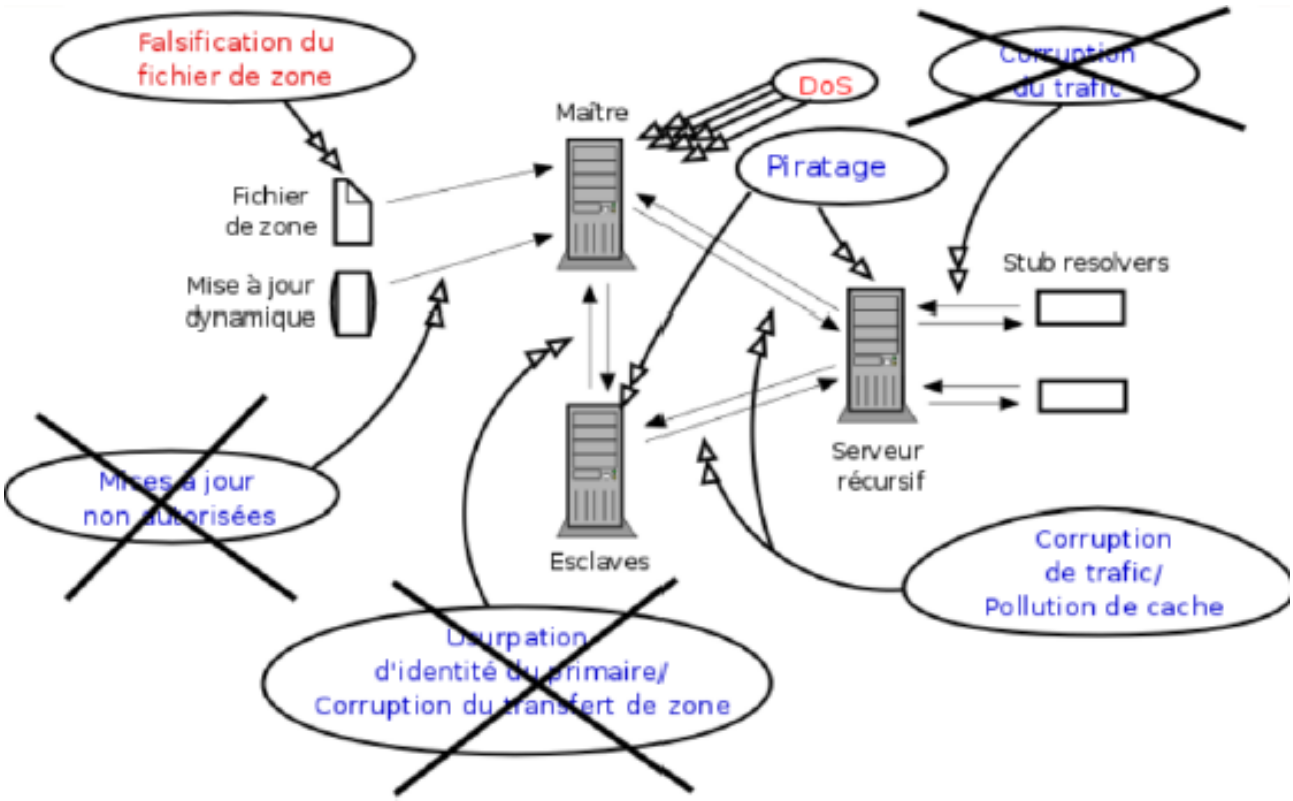

FIGURE 20 – Les attaques éventuellement contrées par DNSSEC. Source : présentation de Stéphane Bortzmeyer à SARSSI 2009.

Néanmoins, l'apport d'une solution à ces attaques n'est pas la priorité de DNSSEC car des solutions existaient déjà pour contrer ces attaques. Les mises à jour dynamiques des zones ou le transfert de zones peuvent être sécurisés par l'intermédiaire de plusieurs méthodes : TSIG, SIG, TKEY, ... Ces méthodes étant encore efficaces, il était inutile de chercher à les remplacer.

Dans la suite de ce rapport, nous parlerons exclusivement des risques propres à DNS, c'est-à-dire au système en lui-même dont le protocole réseau du même nom fait partie intégrante. Nous ne parlerons pas des autres risques techniques attenants à l'usage de DNS.

Nous ne parlerons pas des pannes (matérielles et logicielles) et du besoin de redondance, ni des bureaux d'enregistrement qui se font compromettre, ni des attaques malveillantes qui ne sont pas dues à une faiblesse propre de DNS (comme DNSChanger, par exemple) ni des problèmes de sécurité classiques comme l'ingénierie sociale.

42. Pour une première approche, voir `https://fr.wikipedia.org/wiki/Attaque_de_l'homme_du_milieu`.

Nous ne parlerons pas non plus des risques administratifs : un domaine qui se fait saisir, un oubli de renouvellement, ...

Tous ces risques sont bien réels, peuvent représenter une perte pour un organisme et nécessitent une attention de la part desdits organismes mais ils sortent du cadre de notre étude.

4.3 Faiblesses DNS

4.3.1 Mauvaise configuration

L'idée selon laquelle le protocole DNS est simpliste est répandue. Cela explique sans doute pourquoi on trouve encore un bon nombre de serveurs en production mal configurés (ouverts, réponses du cache accessibles, transfert de zone autorisé à tout le monde, ...).

De plus, la bonne configuration nécessite quelques recherches. Pour illustrer ceci, voici une petite anecdote : dans la première version de nos maquettes, le serveur récursif était fermé et ne répondait qu'à un nombre défini de réseaux. Pourtant, il était possible, depuis les réseaux non autorisés, d'accéder au cache du serveur. La directive « allow-query-cache » permet de résoudre ce problème comme nous l'avons exposé dans la section « Mise en œuvre » de la partie relative à DNS de ce rapport.

DNSSEC ne changera rien à ce constat, ce sera même pire : DNSSEC, en raison de son usage intensif de la cryptographie, nécessite un plus grand soin dans le déploiement et la maintenance régulière d'un serveur DNS (que ce soit un serveur faisant autorité ou un serveur récursif).

4.3.2 Nom de domaine fantôme

Cette faiblesse, découverte au début de l'année 2012, permet de faire conserver, par un serveur cache, un nom de domaine qui a été supprimé de la zone parente et qui ne devrait donc plus pouvoir être résolu.

L'intérêt est tout trouvé pour, par exemple, une activité de phishing ou pour maintenir en activité l'infrastructure de commande et de contrôle d'un botnet ou bien encore pour maintenir des sites censurés à raison ou à tord par des gouvernements ou des agences gouvernementales (FBI aux USA, l'ARJEL ou le gouvernement français via la LOPPSI 2 en France).

Voici la démarche à employer. L'attaquant est ici le possesseur d'un nom de domaine (disons mycorporation.jide.) qui, se faisant supprimer son domaine, tentera de le maintenir dans le cache d'un ou plusieurs résolveur(s). L'attaquant s'occupe lui-même du serveur faisant autorité sur la zone. Pour l'exemple, on considère que ce serveur a pour nom primaire.mycorporation.jide. et pour adresse IP 172.16.0.1.

Tant que le domaine est encore en activité, l'attaquant exécute une requête portant sur un sous-domaine du domaine concerné (exemple : A www.mycoporation.jide. ?). Le serveur récursif va récupérer le nom du serveur faisant autorité sur la zone, l'adresse IP de ce dernier ainsi que

les informations demandées sur l'enregistrement. Tant que le domaine existe, l'attaquant doit rafraîchir régulièrement le cache en exécutant toujours ce même type de requête.

Quand la zone supérieure, jide. dans notre cas, décide d'invalider le nom de domaine en supprimant la délégation, le serveur récursif peut encore résoudre le domaine tant que le TTL des enregistrements concernant la délégation (NS, A) n'a pas expiré. Un peu de temps après cette suppression mais avant que les TTL n'expirent, notre attaquant change le nom du serveur faisant autorité sur la zone dans le fichier de zone (SOA et NS). Fixons le nouveau nom à prim.mycorporation.jide. Ensuite, l'attaquant interroge le serveur récursif sur l'enregistrement A du nouveau NS (exemple : A prim.mycorporation.jide. ?).

Le serveur récursif, possédant déjà les informations sur la délégation, contacte directement le serveur faisant autorité et constate, dans la section « authority » de la réponse, que le nom du serveur faisant autorité a changé. Il met donc à jour les informations de son cache en rafraîchissant le TTL des enregistrements.

La boucle est bouclée. Le domaine (ainsi que tous les enregistrements attenants) reste résolvable pour une nouvelle période. En faisant renouveler régulièrement les informations, l'attaquant permet donc de maintenir en vie son domaine, pourtant inexistant pour le reste de la hiérarchie DNS, dans un ou plusieurs caches, tant que ces caches ne sont pas vidés.

Cette faiblesse ne fonctionne pas sur tous les logiciels serveur DNS car il s'agit d'une faiblesse d'implémentation. Par exemple, la dernière version de Unbound en date au moment de la découverte de ce problème ne laissait pas passer ce comportement alors que la dernière version de BIND le permettait. Ce problème est dû au fait que, les enregistrements NS de la zone fille faisant autorité, certaines implémentations n'iront pas vérifier auprès de la zone parente si la délégation est toujours d'actualité. À la décharge des développeurs des logiciels serveur faillibles, les RFC ne contiennent que très peu d'informations concernant la politique de gestion du cache. Il est donc normal que les auteurs des différentes implémentations aient fait des choix différents.

Pour résoudre efficacement ce problème, il faudrait une concertation à l'IETF concernant la politique de gestion du cache car les solutions à ce problème ne sont pas triviales et nécessitent d'être harmonisée. En attentant, les implémentations faillibles ont été corrigées.

Nous avons réussi à constater l'existence de cette faiblesse dans notre laboratoire Netkit. Nous avons cherché à maintenir le domaine mycorporation.jide. dans le cache de notre serveur récursif. Voici la démarche que nous avons suivie :

1. Dans un premier temps, on demande à notre serveur récursif la résolution du nom www.mycorporation.jide. On vérifie également que le serveur a bien mis en cache les enregistrements NS/A relatif au serveur faisant autorité :

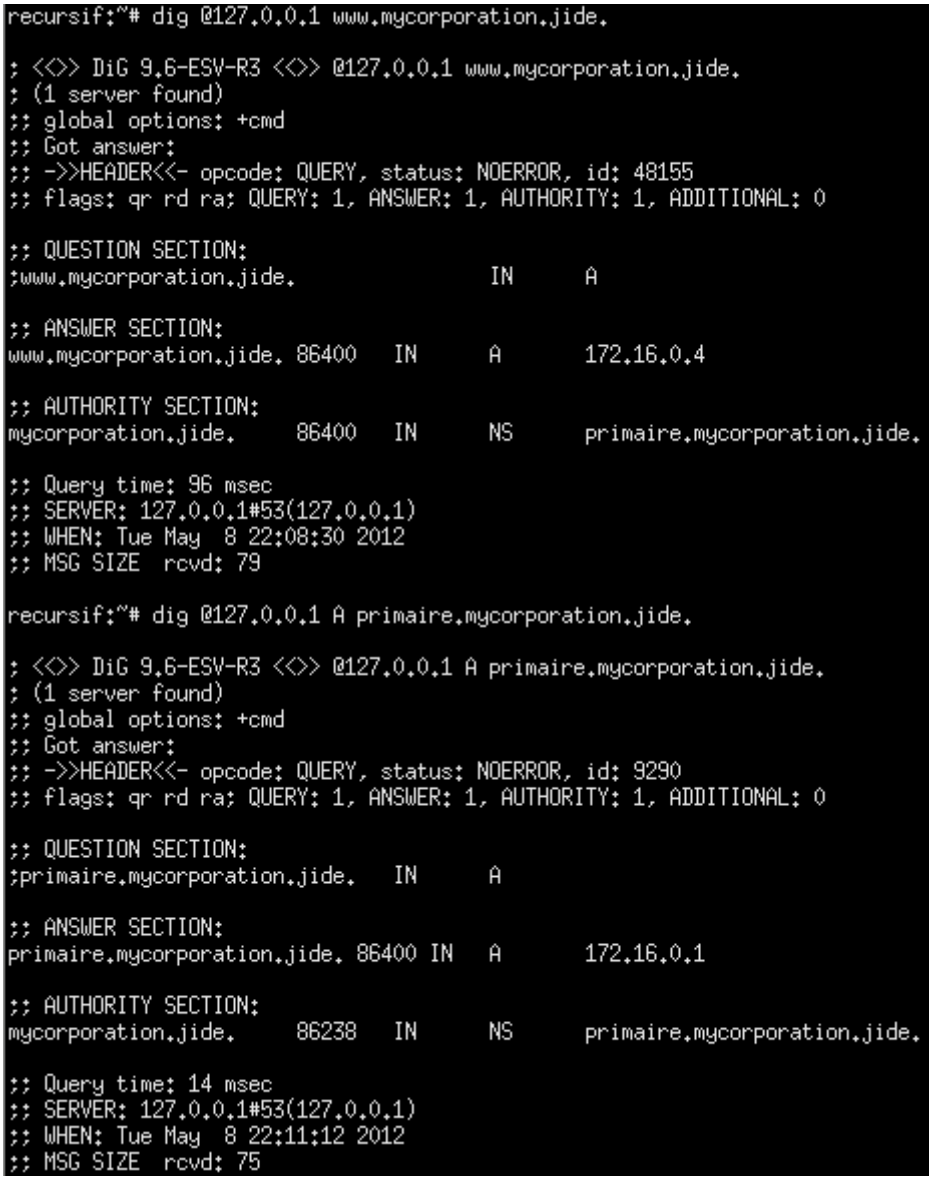

FIGURE 21 – Résolution du nom www.mycorporation.jide par le serveur récursif.

2. Dans un deuxième temps, on supprime le domaine mycorporation.jide de la zone jide :

```
TLD:~# /etc/init.d/bind9 restart
Stopping domain name service...: bind9.
Starting domain name service...: bind9.
TLD:~# dig @127.0.0.1 NS mycorporation.jide.

; <<>> DiG 9.6-ESV-R3 <<>> @127.0.0.1 NS mycorporation.jide.
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 57430
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;mycorporation.jide.            IN      NS

;; AUTHORITY SECTION:
jide.                   900     IN      SOA     ns.jide. hostmaster.jide. 2012022400 3600 1800 604800 3600

;; Query time: 6 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue May  8 22:12:53 2012
;; MSG SIZE  rcvd: 86
```

FIGURE 22 – Suppression du domaine mycorporation.jide. du fichier de zone jide.

3. Ensuite, on change le NS de mycorporation.jide. dans le fichier de zone de la machine Primaire pour le faire passer de primaire.mycorporation.jide. à prim.mycorporation.jide.

```
primaire:~# dig @127.0.0.1 NS mycorporation.jide.

; <<>> DiG 9.6-ESV-R3 <<>> @127.0.0.1 NS mycorporation.jide.
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 47623
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;mycorporation.jide.            IN      NS

;; ANSWER SECTION:
mycorporation.jide.     86400   IN      NS      prim.mycorporation.jide.

;; ADDITIONAL SECTION:
prim.mycorporation.jide. 86400  IN      A       172.16.0.1

;; Query time: 8 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue May  8 22:18:37 2012
;; MSG SIZE  rcvd: 71
```

FIGURE 23 – Modification de la zone mycorporation.jide. pour indiquer le nouveau nom du serveur faisant autorité sur la zone.

4. On demande au serveur récursif de résoudre le nom A prim.mycorporation.jide. On constate que le TTL du serveur faisant autorité a été remis à sa valeur initiale, preuve du bon fonctionnement de notre attaque.

```
recursif:~# dig @127.0.0.1 prim.mycorporation.jide.

; <<>> DiG 9.6-ESV-R3 <<>> @127.0.0.1 prim.mycorporation.jide.
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28686
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 0

;; QUESTION SECTION:
;prim.mycorporation.jide.       IN      A

;; ANSWER SECTION:
prim.mycorporation.jide. 86400  IN      A       172.16.0.1

;; AUTHORITY SECTION:
mycorporation.jide.     86400   IN      NS      prim.mycorporation.jide.

;; Query time: 32 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Tue May  8 22:19:56 2012
;; MSG SIZE  rcvd: 71
```

FIGURE 24 – Résolution du nom prim.mycorporation.jide.

L'impact de cette faiblesse est limité car :

- Il faut toucher un grand nombre de résolveurs de différents réseaux pour espérer obtenir un réel effet.
- Cette faiblesse concerne uniquement certains logiciels serveur dans une version bien définie. Cela limite le nombre de serveurs attaquables à la liste des serveurs non mis à jour.

Cette faiblesse ne peut normalement pas fonctionner sur un serveur récursif dont la validation DNSSEC est activée. En effet, si un résolveur est configuré pour effectuer la validation des enregistrements, il est obligé de vérifier l'intégralité de la chaîne de confiance. Voir la section « Délégation et chaîne de confiance » de la partie « DNSSEC » pour plus d'informations. En tout cas, nous n'avons pas réussi à réaliser cette attaque sur notre maquette DNSSEC.

4.3.3 Empoisonnement de cache

Cette attaque consiste à introduire une réponse falsifiée dans un serveur cache. Le but étant que, lorsqu'un client interrogera le serveur sur le nom falsifié, le serveur lui communiquera la fausse réponse. Cette attaque est certainement la plus redoutée. En effet, un enregistrement « www.banque.fr A 164.81.60.14 » nous permet de rediriger les clients de cette banque sur un faux site web et d'obtenir ainsi des informations intéressantes.

On comprend néanmoins que cette attaque, comme celle des noms de domaine fantômes, a une portée relative au nombre d'utilisateurs du résolveur. Par exemple, le serveur récursif d'Unilim n'est utilisé que par les membres de l'université de Limoges et associés alors que celui de n'importe quel FAI est massivement utilisé. Seules les personnes utilisant un résolveur donné seront impactés lors de l'attaque de celui-ci. De plus, nous comprendrons assez vite que cette attaque se joue de faibles probabilités.

Comment fait-on pour empoisonner un cache DNS ? DNS fonctionne principalement sur le protocole UDP qui est un protocole en mode non connecté. UDP n'a donc pas la notion de sessions et les deux machines qui communiquent ne peuvent pas s'authentifier de manière réciproque. En TCP, les deux machines s'authentifient à travers la phase d'établissement de la communication (le 3-way handshake) et les numéros de séquence qui commencent à un numéro imprédictible. Quand un serveur cache, typiquement récursif, interroge un serveur faisant autorité sur une zone, il ne peut donc pas savoir si la réponse vient bien du bon serveur ou d'une machine qui a forgé un faux paquet de réponse en usurpant l'adresse IP du serveur faisant autorité. Il suffit donc, à un attaquant, de prévoir l'instant où une requête sera émise et d'y répondre. Deux difficultés apparaissent.

La première difficulté est de prévoir l'instant de l'émission d'une requête. Si le serveur récursif est un serveur ouvert (qui répond à tout internet), il suffit d'interroger le serveur sur un nom pour déclencher une requête. L'attaque se complexifie avec un serveur récursif fermé mais elle n'est pas impossible. Imaginons que l'attaquant a le contrôle d'une machine située sur le réseau autorisé à interroger le serveur cache, par exemple ... Imaginons que l'attaquant envoie une quelconque URL à un utilisateur du réseau autorisé ...

La deuxième difficulté est de répondre à la requête. D'une part, il faut répondre plus vite que le vrai serveur faisant autorité. Plusieurs aspects peuvent jouer en la faveur de l'attaquant comme notamment sa proximité, en terme de saut de réseau, du serveur à attaquer. Par exemple : si l'attaquant est capable de se positionner sur un réseau voisin du réseau sur lequel se trouve la machine qu'il souhaite attaquer, il n'aura que très peu de saut pour arriver

à destination en comparaison avec le vrai serveur faisant autorité qui est possiblement à l'autre bout de l'internet. Si ce n'est pas le cas, ou en complément, on peut toujours imaginer que l'attaquant déclenche, en parallèle, une attaque (D)DoS contre le vrai serveur faisant autorité pour se donner du temps et des chances mais cela peut s'avérer difficile sur les grosses zones pour lesquelles des techniques particulières sont mises en place (pare-feu, anycast, ...).

D'autre part, des mécanismes de défense ont été prévus dès la conception du protocole DNS. En effet, le résolveur n'acceptera la réponse que si les informations de la réponse appartiennent bien à la même zone que celle de la question et qu'elles correspondent bien à une question en attente. La réponse devra également être reçue sur le même port que celui qui a servi à l'émission de la question. Mais le principal frein à une attaque par empoisonnement de cache reste le fait que chaque requête contient un numéro de transaction (Query ID, parfois écrit QID ou TXID). Ce numéro aléatoire, codé sur 16 bits doit être identique dans la question et dans la réponse.

Il y a encore peu de temps, le port source était statique sur les principales implémentations de serveur DNS (seuls Dnscache [43], Unbound [44] et PowerDNS [45] faisaient exception). Il ne reste donc qu'à faire en sorte que le serveur à empoisonner génère une requête et à deviner le Query ID. En dehors de problèmes d'implémentation temporaires (ex : en 2007, on découvrait que le générateur de nombre aléatoire de BIND était bogué), ce numéro est imprédictible. Pour avoir la moindre chance, il faut donc envoyer $2^{16} = 65\,536$ paquets forgés au serveur que l'on cherche à empoisonner.

Quand on sait que cette faiblesse est connue depuis longtemps (Paul Vixie, premier auteur de BIND, la décrivait déjà en 1995), on est en droit de se demander comment le DNS peut-il encore fonctionner. D'une part, les réseaux informatiques étaient plus lents que ceux dont on dispose aujourd'hui et empêchaient, de ce fait, l'envoi des 65 536 paquets nécessaires dans l'intervalle réduit. D'autre part, chaque tentative échouée fait que le résolveur va mettre en cache la vraie réponse fournie par le serveur faisant autorité. Le TTL de l'enregistrement reprendra sa valeur initiale et une nouvelle attaque n'est donc pas possible avant l'expiration de ce TTL. En effet, si le serveur est interrogé durant la période de validité du TTL, il donnera la réponse qu'il a en cache et n'effectuera donc aucune requête.

43. Voir : `http://cr.yp.to/djbdns/forgery.html`.
44. Voir : `http://unbound.nlnetlabs.nl/pipermail/unbound-users/2008-July/000133.html`.
45. Voir : `http://blog.netherlabs.nl/articles/2008/07/09/some-thoughts-on-the-recent-dns-\discretionary{-}{}{}vulnerability`.

Jusqu'à l'hiver 2007[46], cette attaque avait donc peu de chances de réussir comme nous venons de l'expliquer. Mais c'est à cette période que Dan Kaminsky découvre une nouvelle méthodologie pour mettre en œuvre cette attaque tout en garantissant un taux de réussite nettement supérieur.

D'une part, au lieu d'interroger le serveur cache sur www.exemple.com., on va l'interroger sur XXXX.exemple.com. où XXXX est un nom choisi aléatoirement. Le serveur cible va donc générer autant de requêtes que de noms demandés. Le QID sera fixe car, vu le nombre de requêtes, une des requêtes générées par le serveur cible aura le même QID que le QID choisi par l'attaquant.

Dans les fausses réponses, l'attaquant ajoutera un RR additionnel contenant des informations sur le nom que l'on souhaite corrompre (www.exemple.com. dans notre cas). Le serveur ne pourra pas refuser ce RR car il fait partie de la zone. Il le prendra donc en considération, le cachera et le resservira à tous les clients qui demanderont www.exemple.com. L'attaque aura réussi.

De plus, les requêtes sont inoffensives : elles correspondent à des noms qui n'existent pas, ne seront pas cachées, et ne bloqueront donc pas la procédure si le serveur faisant autorité répond avant l'attaquent, contrairement à l'attaque classique.

Plusieurs variantes de cette attaque existent et notamment une, implémentée, entre autres, dans Metasploit, qui, plutôt que de corrompre un enregistrement "normal", corrompt un des enregistrements NS de la zone afin de prendre le contrôle sur la zone entière plutôt que sur un seul enregistrement de celle-ci.

Stephane Bortzmeyer, ingénieur à l'AFNIC, assurait, lors des JRES 2009, que les implémentations publiques d'alors permettaient d'empoisonner n'importe quel résolveur en quelques minutes alors que des implémentations restées confidentielles permettaient de le faire en quelques secondes grâce à des optimisations telles que celle apportée par le paradoxe des anniversaires[47].

Pour convaincre de la réalité de cette attaque en dehors des environnements de test, on peut citer, par exemple, le cas de Free, dont l'un des résolveurs mis à la disposition des clients de ce FAI, fournissait une fausse réponse pour le domaine whois.com, suite à un empoisonnement, durant le mois de janvier 2010[48].

46. Bien que la faille fût annoncée publiquement en juillet 2008 (voir : `http://www.kb.cert.org/vuls/id/800113/`), Kaminsky l'avait découverte avant et l'avait déjà annoncée, de manière non publique, aux principaux acteurs impliqués.

47. Pour une première approche, voir : `https://fr.wikipedia.org/wiki/Paradoxe_des_anniversaires`.

48. Voir : `http://www.bortzmeyer.org/empoisonnement-dns-en-vrai.html`.

Nous avons tenté cette attaque en utilisant le module « DNS BailiWicked Host Attack » [49] du framework Metasploit. Le but de ce module est d'empoisonner un enregistrement précis. Dans notre mise en oeuvre, nous essayerons d'associer l'enregistrement A www.mycorporation.jide. à l'adresse IP 42.42.42.42.

En utilisant Metasploit sur la même machine que notre laboratoire Netkit, nous rencontrons un problème de routage : le module Metasploit semble ignorer la table de routage de l'hôte et injecte les fausses réponses sur l'interface réseau principale (eth0) plutôt que sur l'interface TAP qui relie notre hôte au routeur « FAI » de notre maquette Netkit. Nos tentatives de configuration du module (gateway, interface, ...) ont échouées. On pourrait tenter d'utiliser la cible route de la table mangle de netfilter/iptables ou une manipulation avec iproute2 pour rediriger les paquets sur la bonne interface.

En utilisant une machine virtuelle qui exécute Metasploit pour lancer l'attaque, nous avons obtenu un résultat : l'attaque a réussi une fois sur nos quatre essais.

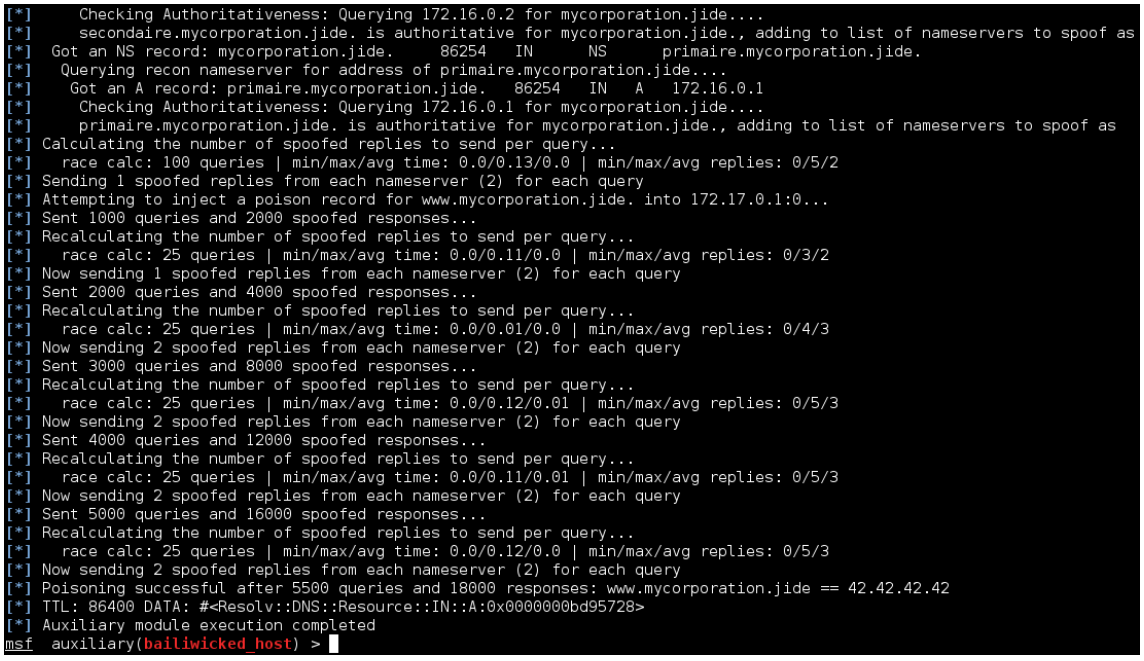

FIGURE 25 – Résultat de notre attaque d'empoisonnement du cache de notre serveur récursif.

49. Voir : `http://www.metasploit.com/modules/auxiliary/spoof/dns/bailiwicked_host`.

On peut fournir plusieurs explications pour ce taux de réussite :

- La probabilité de réussir n'est pas nulle.
- Coup de chance ?
- Nous sommes sur un réseau local. Le résultat serait fortement compromis si nous devions traverser plusieurs réseaux avant d'atteindre le serveur résolveur à empoisonner.
- D'un côté il y a un réseau local entre l'attaquant et le laboratoire Netkit. De l'autre, le serveur faisant autorité sur la zone est une machine Netkit directement reliée au serveur récursif. Cette différence peut expliquer un taux de réussite aussi faible.
- La version de BIND incluse dans le filesystem que nous avons utilisé avec Netkit est 9.6-ESV-R3 [50]. Celle-ci intègre un mécanisme rendant le port source de la requête sortante du résolveur imprévisible. Ainsi, un peu grand nombre de combinaisons est possible et nécessite donc l'envoi de beaucoup plus de requêtes pour mener à bien l'attaque.

Nous n'avons obtenu aucun résultat avec le module « DNS BailiWicked Domain Attack » [51] (qui permet de corrompre un enregistrement de type NS plutôt qu'un enregistrement standard) du framework Metasploit malgré nos quatre essais. Il serait également intéressant d'écrire un programme à partir de rien avec Scapy.

Plusieurs solutions existent à cette faiblesse :

- Utiliser un QID codé sur 128 bits.
- Utiliser uniquement le protocole TCP pour le transport des requêtes DNS.
- Rendre les requêtes moins prévisibles.

Les deux premières solutions ne sont pas envisageables car elles casseraient la compatibilité ascendante que l'on tente de conserver dans tout protocole. De plus, l'utilisation unique de TCP ralentirait la résolution (le temps d'établir la session TCP) et surchargerait les serveurs faisant autorité.

La dernière solution est actuellement celle retenue par un large consensus : on rend le port source de la requête imprévisible (méthode dite « Source Port Randomization »). On passe ainsi de seulement 16 bits d'entropie (le QID) à 32 [52] (le QID + le port source, qui est également codé sur 16 bits). D'autres implémentations sont également possibles. Google, par exemple, utilise, sur ses révoleurs ouverts au public, des requêtes sensibles à la case. Ainsi, si un attaquant tente de corrompre le cache de ces serveurs, il doit trouver le QID, le port source et la case de la

50. Voir : `https://www.isc.org/software/bind/advisories/cve-2008-1447` ou `https://www.isc.org/software/bind/security/matrix`.
51. Voir : `http://www.metasploit.com/modules/auxiliary/spoof/dns/bailiwicked_domain`.

requête. Cette solution était d'ailleurs en discussion à l'IETF mais elle n'a pas rencontré un grand succès.

Néanmoins, cette solution n'est que temporaire. En effet, les vitesses de transmission des réseaux vont encore s'améliorer et rendre à nouveau cette attaque accessible (les serveurs faisant autorité répondront certes plus vite mais un (D)Dos à leur encontre ne sera que plus efficace).

Nous noterons que plus de quatre ans après, des résolveurs sont toujours vulnérables. En effet, selon le rapport sur la résilience de l'Internet français [**?**], 9 à 10 % des résolveurs qui ont interrogé les serveurs faisant autorité pour fr. durant les mois de novembre et décembre 2011 utilisent trop peu de ports sources différents voire, dans le cas le plus commun, n'en utilisent qu'un seul. Ces résolveurs ont été à l'origine de 5 à 8 % des requêtes reçues par les serveurs faisant autorité pour fr. sur la période.

Une dernière solution existe, et elle est efficace : DNSSEC. L'attaquant ne connaissant pas la clé privée de la zone qu'il souhaite attaquer, il ne peut pas forger un enregistrement valide : la signatures ne pourra pas être validée avec les clés publiques de la zone.

52. Pour être complet : le numéro de port est bien codé sur 16 bits mais il faut retirer du lot les ports dits privilégiés, c'est-à-dire les ports 0 à 1023. Il reste donc 2^{16} - 1024 ports. En ajoutant le QID, on arrive donc à (2^{16} - 1024) * 2^{16} combinaisons.

4.3.4 Réflexion/Amplification

On interroge un ou plusieurs serveur(s) récursif(s) en falsifiant l'adresse IP source. Ainsi, le(s) serveur(s) répondra(ont) à l'adresse IP indiquée, qui est celle notre cible. Cette attaque permet, dans le pire des cas, de réaliser un déni de service sur une cible choisie voire également sur le(s) résolveur(s) servant de relais à l'attaque.

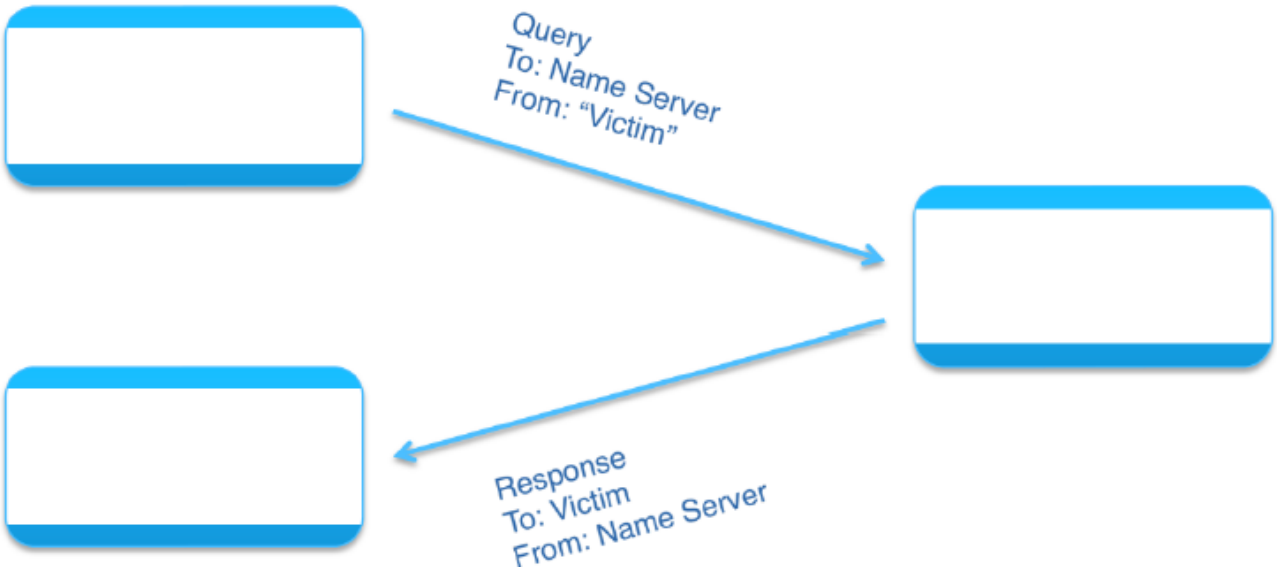

FIGURE 26 – Principe de fonctionnement d'une attaque par réflexion/amplification. Source : présentation réalisée par Duane Wessels lors de la réunion DNS-OARC en octobre 2011.

Cette attaque est intéressante car elle permet de masquer, dans une certaine mesure, l'adresse IP de l'attaquant puisque l'adresse IP qui apparaîtra dans les journaux de la cible ne sera pas celle de l'attaquant mais celle du serveur récursif qui aura servi de relais à l'attaque.

Néanmoins et comme toujours, l'attaque laisse des traces. Ainsi, il est possible de remonter les flux d'opérateur en opérateur mais cela nécessite la coopération de tous les opérateurs ayant participé au transport du flux malveillant.

Duane Wessels a cependant présenté, en octobre 2011 [**?**], une amélioration permettant de retrouver l'opérateur d'où est émis le trafic malveillant en analysant les changements d'instance de destination dans le cas de serveurs DNS fortement anycastés. En effet, BGP [53] change les routes pour s'adapter aux conditions du réseau et l'attaque est donc dirigée vers différentes instances anycastées d'un même serveur en fonction de la nouvelle route. En suivant le principe "ce qui vient du même endroit sera routé de la même façon", il est possible de remonter jusqu'à

l'opérateur source de l'attaque. Cette analyse reste néanmoins un cas d'école car elle nécessite un serveur fortement anycasté et le traitement de journaux conséquents.

Cette attaque n'est pas spécifique à DNS mais à tout protocole fonctionnant au-dessus du protocole UDP car ce dernier permet de communiquer en mode non connecté et ne peut donc pas authentifier les hôtes de manière réciproque. Néanmoins DNS est très intéressant par son ratio taille de la question/taille de la réponse. Une petite question peut, en effet, générer une très grande réponse. D'où le nom de cette attaque « réflexion/amplification ».

Dans un premier temps, l'attaquant doit forger la requête qui obtiendra la plus grande réponse tout en sachant qu'une réponse trop grande sera commutée sur TCP et l'attaque ne sera donc pas possible (car la cible, n'ayant rien demandé, ne se connectera pas au serveur DNS en TCP lorsqu'elle recevra l'indicateur de réponse tronquée en UDP). Il faut utiliser EDNS0, définit dans le RFC 2671, afin de définir la taille maximale du tampon d'entrée. Si la réponse dépasse cette taille, alors le serveur renverra une réponse indiquant que la réponse est tronquée et le client devra utiliser TCP pour obtenir la réponse. Dans la majorité des cas, il ne faut pas que la réponse dépasse 4096 octets (soit 4k).

Deux méthodologies peuvent être appliquées pour obtenir la plus grosse réponse. Soit l'attaquant tente de trouver les plus gros enregistrements d'une zone, soit il a le contrôle d'une zone et ajoute un enregistrement TXT multilignes d'une taille importante à l'intérieur de cette zone.

Pour augmenter l'effet, il faut que la réponse dépasse 1500 octets, c'est-à-dire la taille standard de la MTU sur internet. Ainsi, la machine cible devra rassembler des fragments IP inutilement.

Pour une attaque encore plus efficace, il convient de vérifier que le(s) serveur(s) récursif(s) dispose(nt), au total, d'une plus grande bande passante que la cible.

Nous avons réalisé une version simplifiée et non distribuée de cette attaque dans notre laboratoire Netkit. La machine « pc » usurpera l'adresse IP du serveur Secondaire (172.16.0.2) afin de contacter le serveur récursif (172.17.0.1) et de demander la résolution de l'enregistrement TXT grostxt.mycorporation.jide.

Il existe plusieurs méthodes pour forger un paquet réseau. Nous avons choisi d'utiliser Scapy [54]. Pour installer Scapy sur la machine « pc », il suffit de télécharger la dernière version

53. Pour une présentation simplifiée de ce protocole, voir : `https://fr.wikipedia.org/wiki/Border_Gateway_Protocol`.

sur scapy.net, de décompresser l'archive téléchargée, d'appliquer un patch permettant l'utilisation d'EDNS0 [55] puis d'installer Scapy avec la commande « python setup.py install ». Le script que nous allons utiliser est simple (et pourrait même être amélioré) :

```
#!/usr/bin/python
#coding=utf-8

from scapy.all import *
import random

monPaquet = IP(src="172.16.0.2", dst="172.17.0.1") / UDP(dport=53) /
        DNS(rd=1, qdcount=1, qd=DNSQR(qname="grostxt.mycorporation.jide.",
                qtype="TXT"), ar=DNSOPTRR(edns_bufsize=4096))

while True:
        monPaquet.payload.sport = random.randint(1024,65535)
        monPaquet.payload.payload.id = random.randint(1,65535)
        send(monPaquet, verbose=0)
```

FIGURE 27 – Script utilisant Scapy afin de réaliser notre attaque par réflexion/amplification.

Ce script permet de générer un flux d'environ 24 paquets/secondes représentant 33 méga-octets/secondes non sollicités par la machine Secondaire. Nous remarquons également la taille d'une question, 97 octets au niveau de la couche de liaison ainsi que la taille d'une réponse, 4194 octets toujours au niveau Ethernet. Cela représente donc un ratio d'environ 43. On comprend mieux l'appellation « amplification » de cette attaque.

```
1429 59.718249000 172.17.0.1 172.16.0.2 DNS  1166 Standard query response 0xf76c  TXT
1430 59.836399000 172.17.0.1 172.16.0.2 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=1bf5) [Reassembled in #1432]
1431 59.836430000 172.17.0.1 172.16.0.2 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=1bf5) [Reassembled in #1432]
1432 59.836444000 172.17.0.1 172.16.0.2 DNS  1166 Standard query response 0x751c  TXT
1433 59.968494000 172.17.0.1 172.16.0.2 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=1bf6) [Reassembled in #1435]
1434 59.968503000 172.17.0.1 172.16.0.2 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=1bf6) [Reassembled in #1435]
1435 59.968508000 172.17.0.1 172.16.0.2 DNS  1166 Standard query response 0xf2da  TXT
1436 60.120272000 172.17.0.1 172.16.0.2 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=0, ID=1bf7) [Reassembled in #1438]
1437 60.120283000 172.17.0.1 172.16.0.2 IPv4 1514 Fragmented IP protocol (proto=UDP 17, off=1480, ID=1bf7) [Reassembled in #1438]
```

FIGURE 28 – Exemple de flux réseau engendré par notre attaque.

Comme cette attaque est indépendante de DNS, les solutions immédiatement disponibles sont à rechercher du côté de celles mises en œuvre lors d'un DoS/DDoS : pare-feu avec des règles évitant de surcharger le pare-feu lui-même (retenir toutes les IPs qui participent à un DDoS pour les bloquer est le meilleur moyen de saturer la machine et de participer à l'attaque malgré soi), NIDS (pour la détection), ...

Notons que DNSSEC n'est pas une solution à cette faiblesse. Au contraire, il permet la génération de requêtes capables d'entraîner une réponse plus lourde, de manière plus facile, à cause, entre autres, des signatures.

54. Pour une présentation rapide, voir : `https://en.wikipedia.org/wiki/Scapy`.
55. Le patch est disponible sur le bugtracker du projet Scapy : `http://trac.secdev.org/scapy/ticket/84`.

Parlons rapidement d'EDNS0. Extension Mechanisms for DNS a été normalisé dans le RFC 2671 en 1999. Il permettait de résoudre des problèmes autour de la taille des réponses auparavant limitées à 512 octets.

Dans ce pseudo RR, le client précise la taille maximale qu'il veut/peut recevoir en UDP. Le serveur prend en compte la demande en fonction de sa configuration. En effet, le serveur peut restreindre la taille maximale de la réponse (avec BIND, cela se fait avec la directive edns-udp-size dont le maximum ainsi que la valeur par défaut est 4096 octets). Il est donc inutile de demander une réponse de 8192 octets, par exemple.

Si la réponse (au sens de la section answer, les données des sections authority et additional n'en font pas partie) ne dépasse pas la taille maximale spécifiée par le client et par la configuration du serveur alors ce dernier envoie cette réponse tout en activant le bit TC pour signaler que les sections authority et additional ont été tronquées.

Si la réponse dépasse la taille maximale spécifiée par le client ou par la configuration du serveur alors ce dernier n'envoie pas la réponse mais simplement le bit TC.

Dans les deux cas, le client est libre de redemander les données manquantes en TCP.

Depuis, EDNS0 permet également d'activer le support DNSSEC grâce au bit DO.

4.4 Faiblesses DNSSEC

4.4.1 Parcours de zone avec NSEC

Nous avons déjà expliqué les tenants et aboutissants de cette faiblesse dans notre présentation détaillée de DNSSEC. Nous tenions néanmoins à montrer un cas pratique réalisé "manuellement", c'est-à-dire sans utiliser un logiciel comme ldns-walk.

Dans l'exemple suivant, nous allons parcourir la racine de notre implémentation de test. Voici le déroulement :

- Nous demandons l'enregistrement NSEC associé au label « . » de la zone. Nous savons que le prochain nom de la zone est « arpa. » et que le label « . » existe dans les types NS, SOA, RRSIG, NSEC et DNSKEY.
- Nous demandons l'enregistrement NSEC associé au label « arpa. ». Nous savons à présent que le prochain label sera jide. et comme précédemment, nous obtenons les types existants pour « arpa. »
- Nous faisons de même pour le label ns.
- Nous avons parcouru toute la zone puisque le prochain label est « . » que nous avons déjà obtenu.
- Afin d'obtenir plus d'informations, on demande chaque type de chaque label.

Au final, nous avons la zone au complet : ses enregistrements et les valeurs associées :

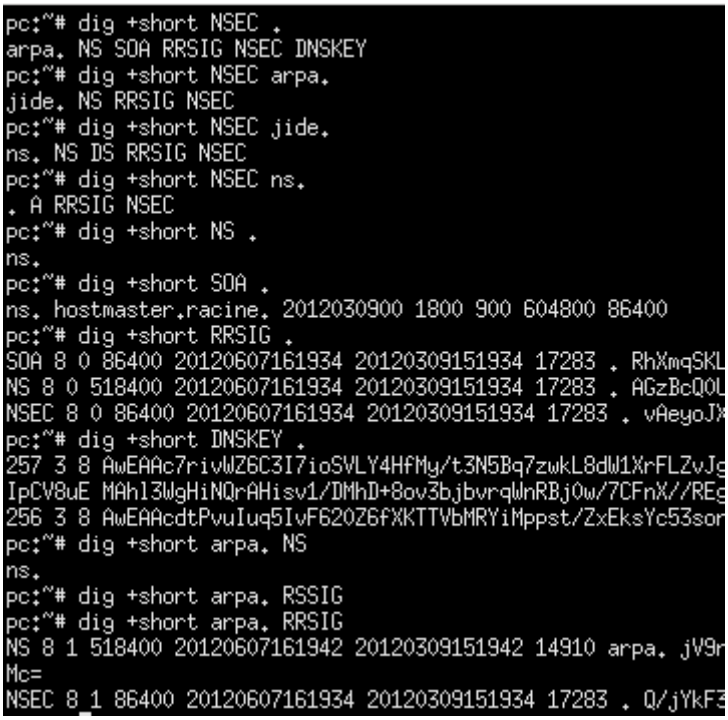

FIGURE 29 – Exemple d'attaque NSEC.

Pour contrer le zone-walking, il est possible d'utiliser NSEC3 lors de la signature de la zone mais si le sel n'est pas changé régulièrement, un attaquant peut très bien générer un dictionnaire ou une rainbow-table qui lui permettrait de retrouver, "en clair", une bonne proportion, voire tout, des FQDN de la zone .

4.4.2 Attaque par rejeu

Comme nous l'avons vu, chaque RRset est signé afin d'assurer son intégrité durant son transport du serveur faisant autorité sur sa zone d'appartenance jusqu'à l'endroit où s'effectue la validation.

La signature des enregistrements se fait, sauf exception, de manière statique : une machine (pas obligatoirement le serveur qui fera autorité sur la zone) signe le fichier de zone à un instant bien précis. Sachant que le QID d'une transaction n'est pas signé, cela signifie qu'il n'y a aucun échange sécurisé d'un nombre imprévisible et aléatoire durant une transaction DNS.

Par contre, les enregistrements contenant les signatures, les fameux RRSIG, contiennent deux dates qui permettent de définir la période de validité de la signature (signature invalide < date de création < signature valide < date de fin < signature invalide).

La non-possesion des clés privées d'une zone donnée ne permet pas de forger un enregistrement avec une signature valide. En revanche, rien n'interdit à un potentiel attaquant de rejouer un RRset et la signature associée durant l'intervalle de validité de cette dernière. En effet, la signature ayant été générée par la partie privée de la ZSK de la zone et son intervalle de validité étant en cours, la signature sera valide donc le rejeu sera accepté par la machine validante.

Cette possibilité de rejeu existait déjà avec DNS "classique" dans lequel rien n'empêche un attaquant d'empoisonner un cache avec une réponse qui était valide autrefois, mais elle était limité : rejouer un enregistrement de type A, AAAA ou NS n'est intéressant que si l'attaquant a aussi le contrôle de la ressource pointée (l'IPv4 d'un A ou le serveur de nom d'un NS), ce qui n'est pas simple (les scénarios envisageables ne dépasseront pas les murs du laboratoire). Plutôt qu'un rejeu incertain, mieux vaut injecter un enregistrement dont on contrôle la ressource pointée.

Certains types d'enregistrement sont néanmoins intéressants à rejouer : CERT, KEY, TLSA, SSHFP, DNSKEY, DS de par leur nature et par le fait qu'ils peuvent être des portes d'entrée à des attaques beaucoup plus préjudiciables et réalisables (man-in-the-middle dans le cas de TLSA, maintien d'enregistrements corrompus dans le cas de DNSKEY/DS, ...).

Notons que le bit de révocation, prévu dans les enregistrements DNSKEY par le RFC 5011, permet la révocation d'une clé utilisée comme trust-anchor par un ou plusieurs résolveur(s) et dont la mise à jour est surveillée par ce(s) dernier(s). Il est inutile dans le cas d'une délégation classique avec des enregistrements DS.

Pour établir un scénario d'attaque, nous allons surtout nous concentrer sur le type TLSA qui permet de stocker soit le certificat d'une autorité de certification X.509, soit un certificat X.509, dans le cas d'une connexion sécurisée par le protocole TLS. Une connaissance minimaliste de DANE [**?**] permet une compréhension plus aisée de ce qui va suivre. Notons bien que d'autres enregistrements (DNSKEY, DS, CERT, SSHFP, ...) peuvent être la cible d'une attaque par rejeu avec un scénario d'attaque et des risques différents.

Pour qu'une attaque par rejeu puisse avoir lieu, il faut un événement déclencheur ainsi qu'une injection de la réponse dépourvue de sens (dans le sens où la donnée injectée n'est plus sémantiquement valable aux yeux de l'administrateur de la zone DNS).

Un événement déclencheur peut être (liste non exhaustive) :

- Une organisation change d'autorité de certification X.509.
- Une organisation change de certificat X.509 (surtout vrai pour les usages 1 et 3 de DANE).
- Une autorité de certification se fait compromettre (usages 0 et 2 de DANE) ou le certificat utilisé se fait compromettre (usages 1 et 3 de DANE).
- Les clés privées de la zone sont compromises et l'attaquant cherche à maintenir une chaîne de confiance alternative qu'il a forgée ainsi que des enregistrements altérés par ses soins.

La probabilité qu'un de ces événements survienne n'est pas nulle. Des compromissions d'autorité de certification X.509 se sont produites tout au long de l'année 2011 [56]. et des changements de certificats voulus par l'organisation elle-même peuvent être fréquents : Google a, par exemple, des certificats d'une durée de vie de 3 jours.

Plusieurs méthodes permettent d'injecter la réponse dépourvue de sens :

- Injecter la réponse dans un résolveur n'utilisant pas DNSSEC (donc vulnérable à l'empoisonnement de cache) et sur lequel un forwarder utilisant DNSSEC vient récupérer les réponses avant de les valider [57]. Notons qu'un serveur utilisant un port aléatoire et DNSSEC ne permet pas d'éliminer totalement le risque. En effet, DNSSEC ne permet d'éviter l'injection que si la signature est invalide (mauvaise clé de signature ou intervalle de validité pas en cours), ce qui n'est pas le cas ici. Rendre le port imprévisible ne fait qu'augmenter l'entropie mais ne rend pas l'injection impossible comme nous l'avons vu déjà vu.

56. En mars 2011, Comodo, autorité de certification X.509, est victime d'un détournement de son "API" et de faux certificats sont émis. Voir `https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html`. En août 2011, c'est au tour de l'autorité de certificat DigiNotar d'être totalement piratée. Plus de 500 faux certificats X.509 sont émis. Voir : `http://sid.rstack.org/blog/index.php/455-parce-qu-il-faut-parler-de-diginotar`. En octobre 2011, l'Electronic Frontier Foundation révèle que d'autres certificats X.509 ont été révoqués avec, comme raison de la révocation « Compromission de l'AC ». Voir : `https://www.eff.org/deeplinks/2011/10/how-secure-https-today`.

– Réaliser une attaque man-in-the-middle entre le résolveur et le client final ou entre le résolveur et internet (si le canal n'est pas sécurisé) afin de remplacer la vraie réponse émanant des serveurs faisant autorité par la réponse dépourvue de sens.

Comme nous l'avons déjà vu, ces attaques ne permettent pas d'avoir une masse critique d'utilisateurs impactés car elles ne passent pas à l'échelle. En effet, si l'on souhaite rejouer le certificat X.509 de Google, il va falloir empoisonner un grand nombre de caches pour avoir un impact fort. Néanmoins, le rejeu d'un enregistrement TLSA permet un man-in-the-middle c'est-à-dire quelque chose d'assez local et ciblé, par définition.

Comme nous venons de le voir, une attaque par rejeu sur DNSSEC n'est pas un "truc de conférencier" pour affoler les masses : la fenêtre d'attaque est limitée, certes, mais le risque n'est pas nul.

On pourrait croire que la solution serait de réduire la durée de vie des signatures mais ce n'est pas le cas car la durée des signatures est intrinsèquement liée à la disponibilité de la zone. Comme nous l'avons vu, la disponibilité d'une zone est surtout assurée par la réplication de celle-ci sur plusieurs serveurs. La période de rafraîchissement de la zone sur les serveurs secondaires additionnée au TTL de l'enregistrement définit la durée de vie des signatures. En effet, si la signature expire avant ce délai, alors les serveurs esclaves pourraient servir des enregistrements dont la signature est expirée.

En comprenant ça, on pourrait être tenté de réduire la durée de la période avant que les secondaires ne rafraichissent la zone mais cela réduirait d'autant la disponibilité de la zone en cas de problème lors du rafraichissement.

Une solution pourrait être de signer en ligne. Cela réduirait l'intervalle durant lequel une attaque par rejeu est réalisable au TTL de l'enregistrement additionné à la dérive de l'horloge. Cette solution n'est pas satisfaisante car elle expose les parties privées des clés et accroît la charge processeur des serveurs faisant autorité.

En 2008, une solution a été proposée par l'Internet Research Lab de l'Université de Californie [**?**] afin de réduire l'intervalle durant lequel le rejeu d'un enregistrement donné est possible pour le ramener à deux fois la durée du TTL de l'enregistrement tout en permettant une signature hors ligne et des durées de vie de signature importantes et donc une résillience accrue de la zone. Pour se faire, le papier propose la création d'un enregistrement RRSIG-H qui reprendrait les informations contenues dans les RRSIG actuellement en usage tout en y ajoutant des condensats

57. L'utilisation de DNSSEC est un pré-requis de DANE, d'où la présence de ce forwarder.

cryptographiques issues d'une chaîne de condensats de Lamport. Le dernier condensat de la chaîne est intégré dans l'enregistrement RRSIG-H et il sera signé. Lors de l'envoi d'une réponse, les serveurs faisant autorité sur la zone n'ont alors qu'à fournir au client DNS l'enregistrement DNS correspondant à la question posée ainsi sa signature associée. Celle-ci contiendra le dernier condensat signé et le condensat "en cours" (non signé). Ce dernier correspondant au X-ième antécédent au condensat signé, tel que X = (date actuelle - date de création de la signature) / TTL. Un client cherchant à valider la réponse n'a alors qu'à calculer X (ou X + 1) fois le condensat du condensat "en cours" et le comparer au condensat signé pour vérifier l'intégrité de la réponse. Pour avoir une visualisation et un exemple concret, je vous renvoie vers le papier détaillant cette solution [**?**].

Cette solution présente néanmoins quelques inconvénients :

- Une chaîne de condensats est lourde à mettre en cache. La longueur de cette chaîne dépend du TTL de l'enregistrement. Dès lors, on comprend qu'il est impossible d'utiliser des TTL courts avec cette méthode.
- L'usage d'une chaîne de condensats nous paraît assez consommateur en stockage et en surcharge CPU même si, d'après les tests réalisés par les auteurs du papier, les effets sont assez limités. Cela nous semble assez peu utilisable sur les grandes zones type TLD.
- La solution réellement présentée dans le papier propose une identification de chaque serveur secondaire faisant autorité sur la zone ce qui ne sert à rien pour atteindre le but recherché.

Une autre solution serait de déléguer les enregistrements sensibles (TLSA notamment) dans une zone spéciale, signée avec une autre paire de clés que la zone principale pour limiter l'impact d'une compromission. La zone serait signée en ligne et l'intervalle durant lequel le rejeu est possible ne pourrait donc pas excéder le TTL de l'enregistrement. Cette solution ne permet néanmoins pas de limiter une attaque par rejeu sur des enregistrements DNSKEY d'une zone principale, ce qui serait utile dans le cas d'une compromission des clés privées d'une zone, par exemple.

5 Conclusion

Dans ce rapport, nous avons présenté le vénérable DNS, ses faiblesses actuelles ainsi que les solutions que tente d'apporter le déploiement actuel des extensions de sécurité de DNS, DNSSEC.

Ce projet d'étude nous a permis de revoir les bases et de parfaire les connaissances que nous avions de DNS. Il nous a également permis de découvrir et de mettre en œuvre, en détail, DNSSEC. Enfin, ce projet nous a donné l'occasion de nous pencher sur la sécurité du DNS, ce que nous n'avions jamais entrepris auparavant.

DNSSEC sera une solution d'avenir, la signature de la racine de l'ICANN et des principaux TLD en 2010 en est une preuve. La sécurité globale du DNS va augmenter sensiblement puisque DNSSEC impose une gestion beaucoup plus rigoureuse des serveurs DNS. Néanmoins, la signature de l'intégralité de la hiérarchie DNS va prendre du temps étant donnée la complexité induite par DNSSEC.

Ce rapport a laissé de côté d'autres problématiques bien actuelles, aussi bien techniques que politiques, du DNS telles que les noms de domaine internationalisés, les racines alternatives ou bien la gouvernance de la racine DNS.

A Annexes

Fichier de zone pour « . »

```
$TTL    86400
$ORIGIN .

@       86400   IN      SOA     ns.     hostmaster.racine (
                                        2012022100
                                        1800
                                        900
                                        604800
                                        86400
                        )


@       518400  IN      NS      ns.
ns      518400  IN      A       10.0.0.1

arpa    172800  IN      NS      ns.
jide    172800  IN      NS      ns.jide.
ns.jide 172800  IN      A       10.0.0.2
```

Fichier de zone pour « arpa ».

```
$TTL    86400
$ORIGIN arpa.

@                       86400   IN      SOA     ns.     hostmaster.racine (
                                                        2012022100
                                                        1800
                                                        900
                                                        604800
                                                        86400
                                                )

@                       518400  IN      NS      ns.

in-addr                 172800  IN      NS      in-addr-srv.arpa.
in-addr-srv.arpa.       172800  IN      A       10.0.0.2
```

Named.conf pour le serveur Racine

```
options {
        listen-on { 127.0.0.1; 10.0.0.1;  };
        directory "/var/named/";
        version "Lucas & Hamza nameserver";

        allow-query { any; };
        allow-query-cache { none; };

        additional-from-auth no;
        additional-from-cache no;

        recursion no;

        allow-transfer{ none; };
        notify no;
};

zone "." IN {
        type master;
        file "zones/master/master.racine";
};

zone "arpa." IN {
        type master;
        file "zones/master/master.arpa";
};

zone "localhost" IN {
        type master;
        file "/etc/bind/db.local";
};
```

```
zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "/etc/bind/db.127";
};
```

Fichier de zone pour « jide. »

```
$TTL    86400
$ORIGIN jide.

@                       900     IN    SOA     ns.jide.        hostmaster.jide. (
                                              2012022400
                                              3600
                                              1800
                                              604800
                                              3600
                                      )


@                       172800  IN    NS      ns.jide.
ns                      172800  IN    A       10.0.0.2

mycorporation           172800  IN    NS      primaire.mycorporation.jide.
mycorporation           172800  IN    NS      secondaire.mycorporation.jide.
primaire.mycorporation  172800  IN    A       172.16.0.1
secondaire.mycorporation 172800 IN    A       172.16.0.2
```

Fichier de zone pour « in-addr.arpa. »

```
$TTL    86400
$ORIGIN in-addr.arpa.

@               900     IN      SOA     ns.jide.        hostmaster.jide. (
                                        2012022400
                                        3600
                                        1800
                                        604800
                                        3600
                                )


@               172800  IN      NS      ns.jide.

1.0.0.10        172800  IN      PTR     ns.
2.0.0.10        172800  IN      PTR     ns.jide.

16.172          172800  IN      NS      primaire.mycorporation.jide.
16.172          172800  IN      NS      secondaire.mycorporation.jide.
```

Named.conf du TLD

```
options {
        listen-on { 127.0.0.1; 10.0.0.2;  };
        directory "/var/named/";
        version "Lucas & Hamza nameserver";

        allow-query { any; };
        allow-query-cache { none; };

        additional-from-auth no;
        additional-from-cache no;

        recursion no;

        allow-transfer{ none; };
        notify no;
};

zone "jide." IN {
        type master;
        file "zones/master/master.jide";
};

zone "in-addr.arpa." IN {
        type master;
        file "zones/master/master.in-addr.arpa";
};

zone "localhost" IN {
        type master;
        file "/etc/bind/db.local";
};
```

```
zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "/etc/bind/db.127";
};
```

Fichier de zone pour « mycorporation.jide »

```
$TTL    86400
$ORIGIN mycorporation.jide.

@       1D      IN      SOA     primaire.mycorporation.jide.    mycorporation.jide. (
                                2012022500
                                3H
                                15M
                                1w
                                3h
                        )

@               IN      NS              primaire.mycorporation.jide.
@               IN      NS              secondaire.mycorporation.jide.
primaire        IN      A               172.16.0.1
secondaire      IN      A               172.16.0.2
@               IN      MX      10      mail.mycorporation.jide.

mail            IN      A               172.16.0.3
www             IN      A               172.16.0.4
web             IN      CNAME           www.mycorporation.jide.

pc1             IN      A               172.16.0.10
pc2             IN      A               172.16.0.11

grostxt         IN      TXT             ( "NWjmyV[...]jYg" )
```

Fichier de zone pour « 16.172.in-addr.arpa »

```
$TTL    86400
$ORIGIN 16.172.in-addr.arpa.

@       1D      IN      SOA     primaire.mycorporation.jide.    mycorporation.jide. (
                                2012022500
                                3H
                                15M
                                1w
                                3h
                        )

@               IN      NS      primaire.mycorporation.jide.
@               IN      NS      secondaire.mycorporation.jide.

1.0             IN      PTR     primaire.mycorporation.jide.
2.0             IN      PTR     secondaire.mycorporation.jide.
3.0             IN      PTR     mail.mycorporation.jide.
4.0             IN      PTR     www.mycorporation.jide.

10.0            IN      PTR     pc1.mycorporation.jide.
11.0            IN      PTR     pc2.mycorporation.jide.
```

Named.conf pour le serveur Primaire

```
options {
        listen-on { 127.0.0.1; 172.16.0.1;  };
        directory "/var/named/";
        version "Lucas & Hamza nameserver";

        allow-query { any; };
        allow-query-cache { none; };

        additional-from-auth no;
        additional-from-cache no;

        recursion no;

        allow-transfer{ none; };
        notify no;
};

// Instructions pour les échanges master<->slave
include "/etc/bind/tsig.conf";

zone "mycorporation.jide" IN {
        type master;
        file "zones/master/master.mycorporation.jide";
        allow-transfer { 172.16.0.2; };
        notify yes;
};

zone "16.172.in-addr.arpa" IN {
        type master;
        file "zones/master/master.172.16.rev";
        allow-transfer { 172.16.0.2; };
        notify yes;
```

```
};

zone "localhost" IN {
        type master;
        file "/etc/bind/db.local";
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "/etc/bind/db.127";
};
```

tsig.conf du serveur primaire

```
key "TRANSFERT" {
        algorithm hmac-sha256;
        secret "fVHHGLEwSJTRUIa6PgIhJmYz4Vcim11QlgInnCUfEIw=";
};

server 172.16.0.2 {
        keys {
                TRANSFERT;
        };
};
```

Named.conf du serveur secondaire

```
options {
        listen-on { 127.0.0.1; 172.16.0.2;  };
        directory "/var/named/";
        version "Lucas & Hamza nameserver";

        allow-query { any; };
        allow-query-cache { none; };

        additional-from-auth no;
        additional-from-cache no;

        recursion no;

        allow-transfer{ none; };
        allow-notify { none; };
        notify no;
};

include "/etc/bind/tsig.conf";

zone "mycorporation.jide" IN {
        type slave;
        file "slave/slave.mycorporation.jide";
        masters { 172.16.0.1; };
        allow-notify { 172.16.0.1; };
};


zone "16.172.in-addr.arpa" IN {
        type slave;
        file "slave/slave.172.16.rev";
        masters { 172.16.0.1; };
```

```
        allow-notify { 172.16.0.1; };
};

zone "localhost" IN {
        type master;
        file "/etc/bind/db.local";
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "/etc/bind/db.127";
};
```

tsig.conf du serveur secondaire

```
key "TRANSFERT" {
        algorithm hmac-sha256;
        secret "fVHHGLEwSJTRUIa6PgIhJmYz4Vcim11QlgInnCUfEIw=";
};

server 172.16.0.1 {
        keys { TRANSFERT; };
};
```

Named.conf du serveur recursif

```
options {
        listen-on { 127.0.0.1; 172.17.0.1; };
        directory "/var/named/";
        version "Lucas & Hamza nameserver";

        allow-query {192.168.1.0/24; 172.16.0.0/16; 172.17.0.0/16; 127.0.0.1/32;
                     192.168.4.0/24; };

        allow-recursion {192.168.1.0/24; 172.16.0.0/16; 172.17.0.0/16; 127.0.0.1/32;
                         192.168.4.0/24; };

        allow-query-cache {192.168.1.0/24; 172.16.0.0/16; 172.17.0.0/16; 127.0.0.1/32;
                           192.168.4.0/24; };

        allow-transfer{ none; };
        notify no;
};

zone "." IN {
        type hint;
        file "/etc/bind/db.root";
};

zone "localhost" IN {
        type master;
        file "/etc/bind/db.local";
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "/etc/bind/db.127";
};
```

db.root du serveur recursif

```
.       3600000 IN      NS      ns.
ns.     3600000 IN      A       10.0.0.1
```

Named.conf du serveur box

```
options {
        listen-on { 127.0.0.1; 192.168.1.254;  };
        directory "/var/named/";
        version "Lucas & Hamza nameserver";

        forwarders { 172.17.0.1; };
        forward only;

        allow-query { 192.168.1.0/24; 127.0.0.1; };
        allow-recursion { 192.168.1.0/24; 127.0.0.1; };
        allow-query-cache {192.168.1.0/24; 127.0.0.1; };

        allow-transfer{ none; };
        notify no;
};

zone "localhost" IN {
        type master;
        file "/etc/bind/db.local";
};

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "/etc/bind/db.127";
};
```

Table des figures

Table des matières