# Migration du serveur DNS Bind sur TinyDNS

Sylvain Roschi

31 janvier 2013

# Sommaire

# Liste des tableaux

# Table des figures

# 1 Résumé

Pour naviguer sur internet nous avons besoin d'un système traducteur du langage humain en langage informatique, car il serait très fastidieux de se rendre sur un site web avec l'adresse http://129.194.184.80:80/ plutôt que http://www.tdeig.ch/. Cette transformation d'adresse est réalisée par le service DNS de l'anglais "Domain Name System" et traduisible par "système de noms de domaine" défini par des nomes RFC (RFC6195, RFC1035, RFC1591).

Il existe plusieurs serveurs DNS dons les plus connus NSD (serveur d'autorité, UNIX), BIND (multiplateforme) et Microsoft DNS (Principalement utilisé dans les réseau interne en liaison avec Active Directory)[3].

Dans le cadre du laboratoire nous utilisons actuellement BIND sur une base Debian.

La migration vers un nouveau serveur DNS *TinyDNS[10]* est étudiée dans ce travail. Nous verrons comment fonctionne TinyDNS ainsi que sa configuration et comment le tester.

# 2 Énoncé

Ce travail comprend 2 parties :

1. Préparation et test du serveur DNS TinyDNS en vue de la migration du serveur DNS (Bind) actuellement en production pour le domaine tdeig.ch du laboratoire.
2. Créer une installation personnalisée de la distribution PfSense[6].

Définition du cahier des charges :

Étudier le serveur DNS TinyDNS et ses mécanismes de sécurités. Installer un environnement de test qui nous permettra de configurer le serveur TinyDNS sur une distribution PfSense et effectuer des tests de sécurités.
La deuxième partie consiste à personnaliser une installation de la distribution PfSense au travers d'un fichier de réponse automatique afin d'obtenir un environnement opérationnel après une installation.

## 2.1 Environnement de laboratoire

Pour le laboratoire j'ai utilisé les deux machines physiques suivantes.

Le premier PC *A29* a servit de client sous Windows 7. Il est équipé d'un processeur Intel® Core™2Duo E8400 à 3GHz et de 8Go de RAM.

Le deuxième PC *A32* a servit de serveur. Il est équipé d'un processeur Intel® Core2™Duo E8400 à 3GHz, 4Go de RAM et d'une carte double interface Ethernet Giga Intel®. Une carte réseau supplémentaire est obligatoire, car la carte réseau intégrée à la carte mère n'est pas reconnue par pfSense.

Le réseau est connecté sur l'interface *A* de la carte additionnelle et nommée *em0* par pfSense. Aucune autre interface n'est utilisée, car les fonctions de Firewall et de routage ne seront pas implémentées dans ce travail.

## 2.2 Réseau

Afin de pouvoir effectuer facilement des analyses au travers de WireShark (depuis le Laptop), les machines (A32, A29 et Laptop) sont relier à un HUB Ethernet. Le HUB mis à disposition supporte un débit de 10Mbps, ce débit est amplement suffisant pour effectuer des tests DNS entre deux machines.

Voici le schéma représentant le réseau [1]

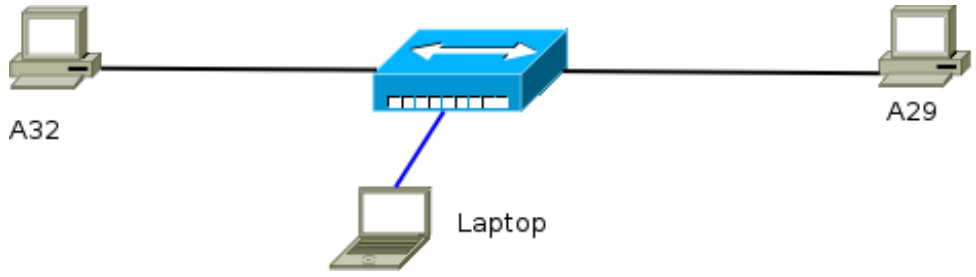



Fig. 2.1 – Schéma du laboratoire

[1] Le lien bleu signifie qu'il est temporaire et non obligatoire, il a existé seulement pour la capture WireShark

Les postes ont étés configurer avec les paramètres réseau suivantes :

| PC | Adresse IP | Masque | Gateway | Port physique |
|---|---|---|---|---|
| A32 | 192.168.1.1 | 255.255.255.0 | 192.168.1.1 | A |
| A29 | 192.168.1.2 | 255.255.255.0 | 192.168.1.1 | carte mère |

Tab. 2.1 – Distribution des adresses IP

## 2.3 Domaine

Configuration DNS utilisé :

| Type | FQDN | Adresse IP |
|---|---|---|
| Domaine, SOA | rshepia.ch | |
| Serveur DNS | ns1.rshepia.ch | 192.168.1.1 |
| Name | sylvain.rshepia.ch | 192.168.1.2 |
| Name | admin.rshepia.ch | 192.168.1.3 |

Tab. 2.2 – Relation FQDN et adresse IP

# 3 Analyse

TinyDNS est le serveur DNS principal de la suite d'outils Djbdns créé par Daniel Julius Bernstein (abrégé par Djb). Convaincu par la conception de ses produits, le créateur propose 1000$ au premier qui y trouve une faille de sécurité[1]. Ce prix a été remporté par Matthew Dempsky en mars 2009 qui a découvert un bogue dans le programme axfrdns[2].

Par défaut cette suite d'outils est résistante à la vulnérabilité du 8 juillet 2008 (CERT VU#800113[9]) Cette faiblesse qui a touché de nombreux serveurs DNS, dont les plus grands (BIND, Microsoft et autres), consiste en une attaque par intoxication du cache (*cache-poisonning*) DNS à la suite d'une erreur d'implémentation du protocole DNS.

Plus concrètement, il s'agit de deux vulnérabilités. La première se trouve à la hauteur de l'ID de transaction généré aléatoirement sur 16bits. Certaines configurations utilisent un ID plus court, il est donc plus facilement possible de le prédire et de l'utiliser pour une attaque. La deuxième concerne une mauvaise implémentation de la gestion des requêtes simultanées. Si le serveur reçoit plusieurs requêtes identiques pour le même enregistrement (RR[1])

## 3.1 Outils Djbdns

Pourquoi Djbdns est-il divisé en plusieurs programmes ? La raison est simple, la sécurité.

Si l'on fractionne un programme en plusieurs binaires, on ajoute une couche d'isolement entre les différents services proposés par ces sous-programmes. Par exemple, un serveur DNS s'occupe de plusieurs tâches, il est composé d'un résolveur, d'un cache, d'une base de données et il peut aussi résoudre des adresses IP en FQDN ou gérer un transfère de zone. Toutes ces applications sont suffisamment détachées les unes des autres pour en faire des programmes différents. Cette conception rejoint la philosophie d'UNIX, d'après Douglas Mcllroy[7] : *ne faire qu'une seule chose, et la faire bien.*

De plus, si l'un de ces services et infecté, il sera plus dur pour un pirate de contaminer les autres services. L'on pourrait aussi imaginer un renfort de sécurité au travers de SELinux en définissant des contextes différents pour chacun des programmes.

La suite de programmes formant le serveur Djbdns se compose principalement des applications suivantes :

### 3.1.1 dnscache

Il s'agit du résolveur de requêtes récursives et d'un cache DNS pour les requêtes provenant des clients. Seules les données provenant des serveurs faisant autorité seront utilisées. Ces derniers sont identifiés par une chaine de délégation provenant des serveurs root[2] Si ce mécanisme d'identification n'existait pas, dnscache serait aussi victime de cache-poisonning.

### 3.1.2 tinydns

Ce programme endosse la tâche de serveur d'autorité pour un ou plusieurs domaines. Il gère les requêtes UDP et ne s'occupe pas des requêtes récursives. Son rôle principal est de communiquer avec les autres serveurs de nom et non de répondre aux requêtes des clients, c'est pour cela que les *best practice* (bonne pratique) recommande de ne pas faire pointer le fichier */etc/resolv.conf* sur un serveur TinyDNS.

[1] Resource Records

[2] Les serveurs root sont les serveurs DNS faisant autorité à la racine , il y en a 13 sur tout le globe. Ils sont identifiés par les lettres de A à M.[5]

### 3.1.3 axfrdns

Ce dernier s'occupe de gérer les requêtes TCP et du transfert de zone.

## 3.2 TinyDNS

Dans ce travail seul l'outil TinyDNS nous intéresse car nous n'utiliserons pas de cache DNS, ni la fonction de forwarding ou de transfère de zone et nous ne ferons pas de résolution des requêtes récursives.

La configuration des zones et des entrées DNS se trouve dans le fichier */var/etc/tinydns/root/data*[3] qui une fois compilé avec l'outil *tinydns-data* nous donne le fichier de la base de donnée */var/etc/tinydns/root/data.cdb*.

Dans une configuration standard (implémentation non PfSense) il nous est possible de configurer une zone DNS aussi que des FQND, soit au travers de commandes, soit en modifiant le ficher */var/etc/-tinydns/root/data* et en y ajoutant la configuration désiré. Ce n'est pas le cas dans ce système d'exploitation car le paquet TinyDNS utilisé contient des scripts PHP permettant l'intégration complète dans l'interface de configuration web et le stockage de la configuration dans le fichier maitre *config.xml*.

Les points §3.2.1 et §3.2.2, sont présentés et expliqués afin de comprendre le fonctionnement de TinyDNS mais ne seront pas utilisé pour la configuration car ce sont les scripts PHP qui s'en chargerons à notre place.

### 3.2.1 Fichier de configuration *data*

Il faut cependant comprendre la syntaxe du fichier *data*[8] pour interpréter la configuration créé par l'environnement PfSense et pour pouvoir la valider.

Les différents enregistrements sont devancés par un symbole définissant le type de l'enregistrement :

| Symbole | Définis | Type | Exemple |
|---|---|---|---|
| . | La zone DNS[4] | SOA | *.tdeig.ch :ns1.tdeig.ch* |
| & | Un serveur de nom | NS | *tdeig.ch : :ns1.tdeig.ch* |
| @ | Un serveur de messagerie | MX | *@hesge.ch :160.53.250.155 :mxsmtp.etat-ge.ch* |
| + | Un hôte | A | *+www.tdeig.ch :129.194.184.80* |
| ^ | L'inverse de A | PTR | *^www.tdeig.ch :129.194.184.80* |
| = | Un hôte et son inverse | A et PTR | *=www.tdeig.ch :129.194.184.80* |
| C | Un alias | CNAME | |
| *.domaine | Un masque d'un domaine | | **.tdeig.ch :129.194.184.80* |

Tab. 3.1 – Exemple des configurations possible dans le fichier data pour TinyDNS

Pour que les changements apportés au fichier *data* soit effectif, il faut le compiler afin qu'ils sont intégrés à la base de données active *data.cdb*. Cette action est fait par l'outil *tinydns-data* généralement utilisé au travers d'un Makefile[5]

[3] Les chemins mentionnés dans ce documents sont valable pour l'OS PfSense, ils peuvent déférer en fonction de leur installation sur un autre système.

[5] */var/etc/tinydns/root/Makefile*

### 3.2.2 Commandes TinyDNS

Il est tous de fois plus facile, et conseillé, d'ajouter des entrées à l'aide des outils suivantes :

| **Commande** | **Définis** |
|---|---|
| /var/etc/tinydns/root/add-ns | l'ajout d'un enregistrement NS |
| /var/etc/tinydns/root/add-alias | l'ajout d'un enregistrement CNAME |
| /var/etc/tinydns/root/add-host | l'ajout d'un enregistrement A |
| /var/etc/tinydns/root/add-mx | l'ajout d'un enregistrement MX |
| /var/etc/tinydns/root/add-childns | l'ajout d'une zone fille |

Tab. 3.2 – Commandes TinyDNS

## 3.3 Particularité du paquet Djbdns dans PfSense

Comme mentionné plus haut (§3.2) dans ce document, Djbdns est intégré dans PfSense afin d'apporter une gestion du service par l'interface web dédié.

Le paquet *TinyDNS* pour PfSense contient, en plus des outils Djbdns, les scripts de configurations et les fichiers XML contenant les pages affichées par la console web et les actions effectuées par ces dernières.

Les outils Djbdsn se trouvent dans le dossier */var/etc/tinydns/* tandis qu'une partie des scripts se trouvent dans le dossier */usr/local/pkg/*.

J'ai put définir une partie des fichiers utilisée pour la gestion de l'interface mais il aurait été trop fastidieux de tous les décrires car il faudrait décortiquer le code source, ce qui aurait pris trop de temps et n'aurais aporté aucune plus value.

Voici les fichiers principaux que j'ai identifiés :

| **Fichier** | **Utilité** |
|---|---|
| tinydns.inc | Fichier principale contenant les fonctions PHP appelant les outils TinyDNS, pour effectuer toutes les actions disponibles (configuration, gestion des logs ou affichage des entrées) |
| tinydns.xml | Contient la structure de la page de configuration (onglet *Settings*) |
| tinydns_domains.xml | Page permettant de créer une nouvelle entrée DNS |
| tinydns_parse_logs.php | Page affichant les logs DNS |
| tinydns_up.php et tinydns_down.php | Créer ou supprimer la configuration dans le ficheir *config.xml* |

Tab. 3.3 – Fichiers de gestions de l'interface Web pour TinyDNS

# 4 Réalisation

## 4.1 Installation du poste A29, client Windows 7

L'installation du poste client sous Windows 7 doit se faire depuis le réseau de la salle de cours afin de bénéficier de l'installation depuis le réseau.

- Démarrer le poste A29.
- Lors de l'affichage du menu proposé sur le réseau choisir le deuxième choix *WinPE* et pressez la touche *[ENTER]*.
- Après quelques secondes l'invité de commande Windows apparait, il faut à ce moment saisir la commande *i 1g2* qui a pour effet d'installer Windows 7 sur le disque local. Cette opération prend environ 2 minutes avant de redémarrer.
- Pendant le redémarrage, débranchez le câble réseau du Switch et branchez le sur le HUB du laboratoire de tests.
- Une fois le poste redémarrer sélectionner *Disque_Dur* ou attendez 30 seconds pour lancer Windows et connectez vous avec l'utilisateur *albert* et le mot de passe *admin*.
- Windows demande une activation, répondez en cliquant sur *Ask me later* puis *ok*[1].

## 4.2 Configuration réseau du poste client

- Ouvrir le panneau de configuration : bouton *Start* puis *Control Panel*
- Selectionner *Network and Sharing Center*
- Dans la partie de gauche selectionner *Change adapter settings*
- Faire un clique droit sur la carte *Local Area Connection* et choisir *Properties*
- Placer le focus sur la ligne *Internet Protocol Version 4 (TCP/IPv4)* puis cliquer sur le bouton *Properties*.
- Une nouvelle fenêtre s'ouvre remplir les champs comme suit :

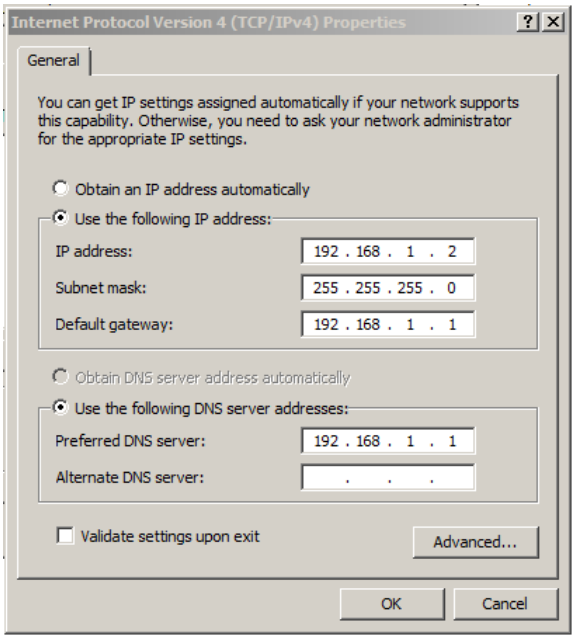



Fig. 4.1 – Configuration IPv4 du poste client

- Sélectionner la localisation du réseau : *Work Network*

[1] N'ayant plus d'accès à internet, il est impossible d'activé Windows

## 4.3 Installation du poste A32, serveur PfSense

- touche *[F8]* > choisir le lecteur CDROM (CDROM :3S-LITE-ON DVDRW LH-20A1S) pour pouvoir démarrer sur le CD
- choix de boot par défaut *1)*

```
Do you want to set up VLANs now[y|n]? n
```

```
Enter the WAN interface name or 'a' for auto-detection: em0
```

```
Enter the LAN interface name or 'a' for auto-detection
NOTE: this enables full Firewalling/NAT mode.
```

tapez sur la touche *[ENTER]*

```
The interfaces will be assigned as follows:
WAN -> em0
Do you want to proceed [y|n]? y
```

- Actuellement aucune Adresse IP n'a été délivrer à l'interface WAN qui est en mode DHCP par défaut
- On choisit l'option *2* pour configurer l'adresse IP de l'interface

```
Configure WAN interface via DCHP? [y|n]
> n
```

```
Enter the new WAN IPv4 address. Press <ENTER> for none:
> 192.168.1.1
```

```
Subnet masks are entered as bit counts (as in CIDR notation) in pfSense.
e.g. 255.255.255.0 = 24
     255.255.0.0   = 16
     255.0.0.0     = 8

Enter the new WAN IPv4 subnet bit count:
> 24
```

```
Do you want to enable the DHCP server on WAN? [y|n] n
Disabling DHCPD...Done!
```

```
Do you want revert to HTTP as the webConfigurator protocol? (y/n) n
```

- Pressez la touche *[ENTER]*
- On choisit l'option *99* pour faire l'installation
- Sélectionner < *Change Keymap (default)*> et choisir *swissfren.iso.acc.kbd* pour changer le clavier
- Sélectionner < *Accept these Settings* >
- Sélectionner < *Quick/Easy Install* > puis < *OK* >
- Sélectionner < *Symmetric multiprocessing kernel (more than one processor)* >. Nous pouvons activer cette configuration du kernel, car l'installation est effectuée sur une machine avec un processeur multicoeur.
- L'installation est terminée, il faut redémarrer afin de travailler sur la bonne instance.

## 4.4 Installation du 'package' Djbdns

– Se connecter à l'interface web au travers d'un navigateur web avec l'URL https://192.168.1.1/
– L'utilisateur par défaut est *admin* et son mot de passe est *pfsense* [2]
– Se rendre dans le gestionnaire de paquet dans le menu *System* > *Packages*.

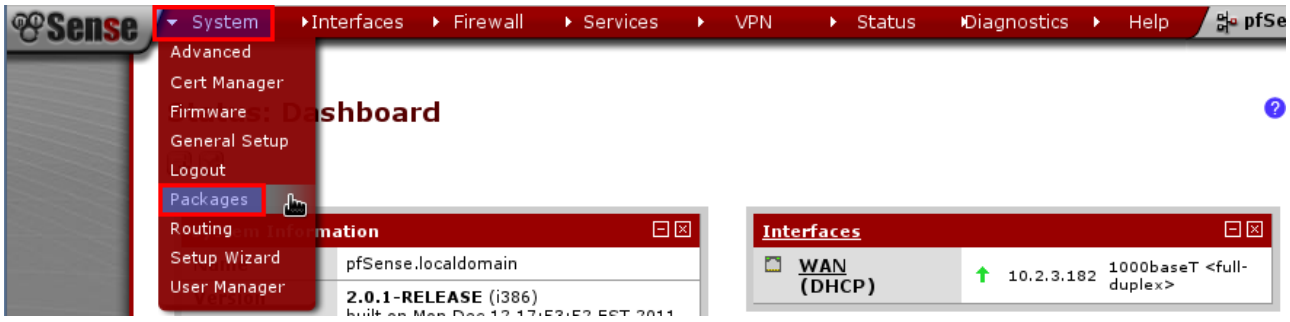



Fig. 4.2 – Menu Gestionnaire de paquets

– Par défaut, on arrive sur les paquets installés, il faut sélectionner l'onglet *Available Packages*


Fig. 4.3 – Gestionnaire de paquets

– Rechercher la ligne comportant le *Package Name dns-server* et cliquer sur le bouton de droite pour ajouter le paquet.

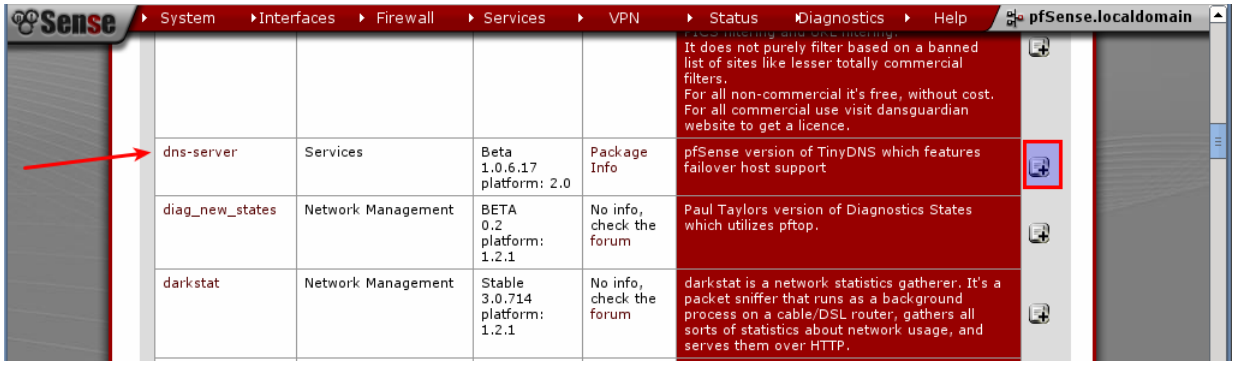



Fig. 4.4 – Sélection du paquet TinyDNS

[2] J'ai gardé le mot de passe par défaut dans l'environnement du laboratoire pour une question de simplicité.

– On peut constater la bonne installation à l'aide du log généré.

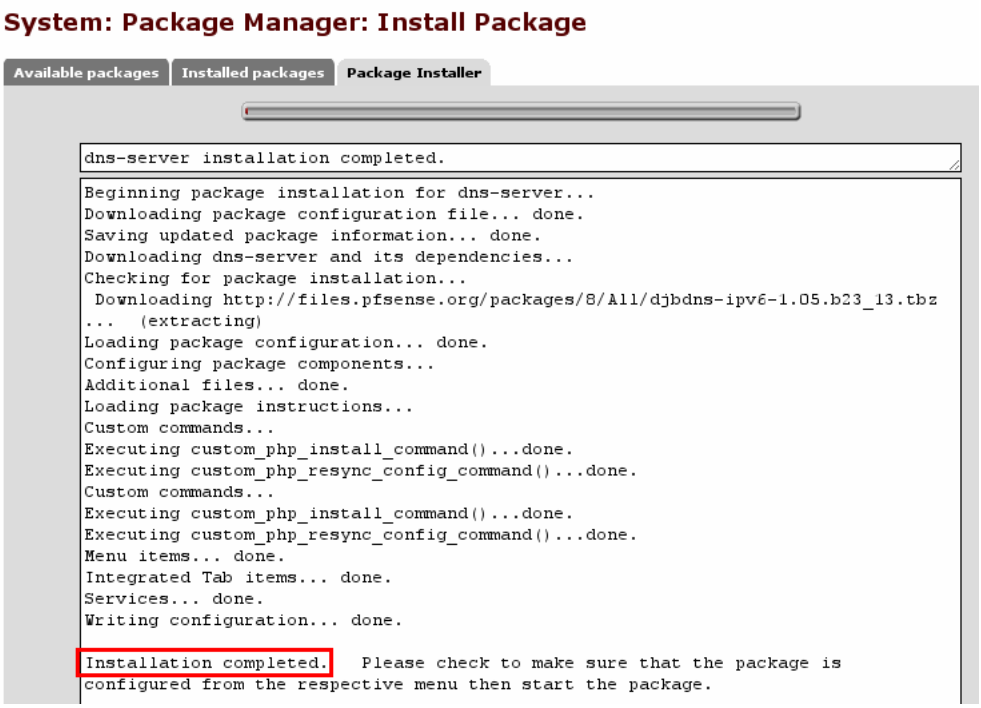



Fig. 4.5 – Résultat installation du paquet

– Constat : le service n'est pas démarré et ne veut pas démarrer. Il démarre après un redémarrage du serveur.

## 4.5 Configuration

Deux possibilités s'offrent à nous, soit au travers de l'interface web, soit en modifiant directement le fichier de configuration.

### 4.5.1 Configuration en mode graphique

– Pour accéder à l'interface web, il faut se rendre à l'adresse https:\\192.168.1.1.

– Les paramètres du serveur DNS se accessible depuis le menu *Status>DNS Server*.

– La première étape consiste à créer le domaine. Cliquer sur l'onglet *wizard* et remplissez les champs *Domain Name* et *Primary Nameserver* :

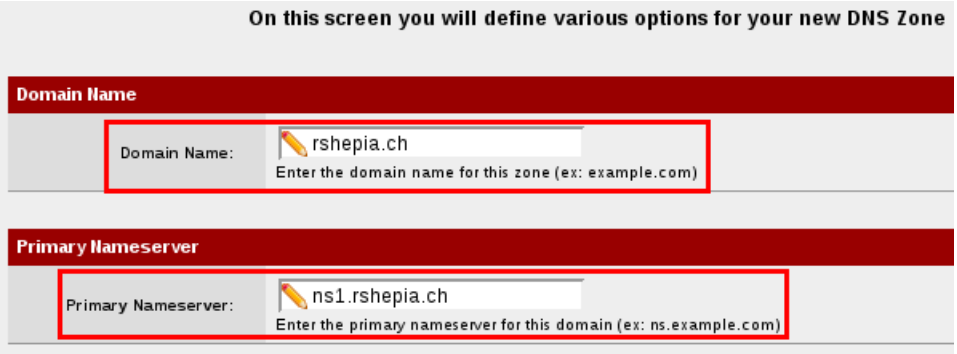



FIG. 4.6 – Wizard, création nouveau domaine

– Il faut définir l'adresse IP du serveur DNS depuis l'onglet *Settings*

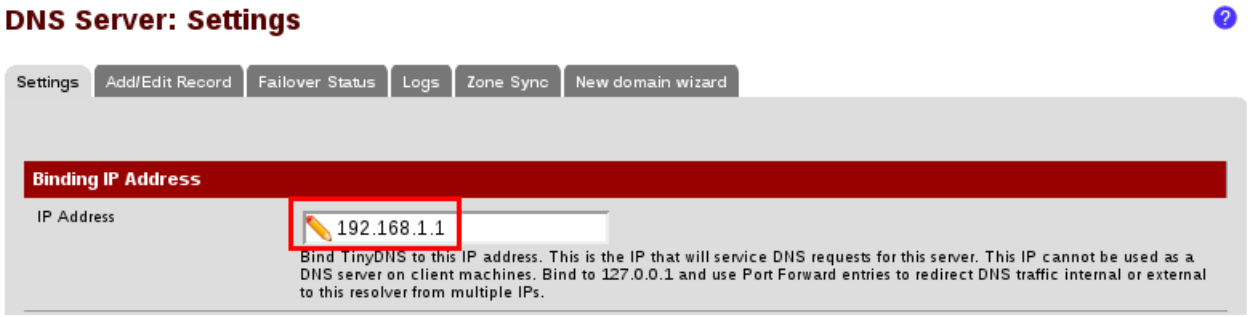



FIG. 4.7 – *Settings*, Adresse IP du serveur DNS

– Créons notre premier enregistrement, celui du NS, en se rendant sur l'onglet *Add/Edit Record* puis sur le petit logo avec le signe +, en bas à droite.

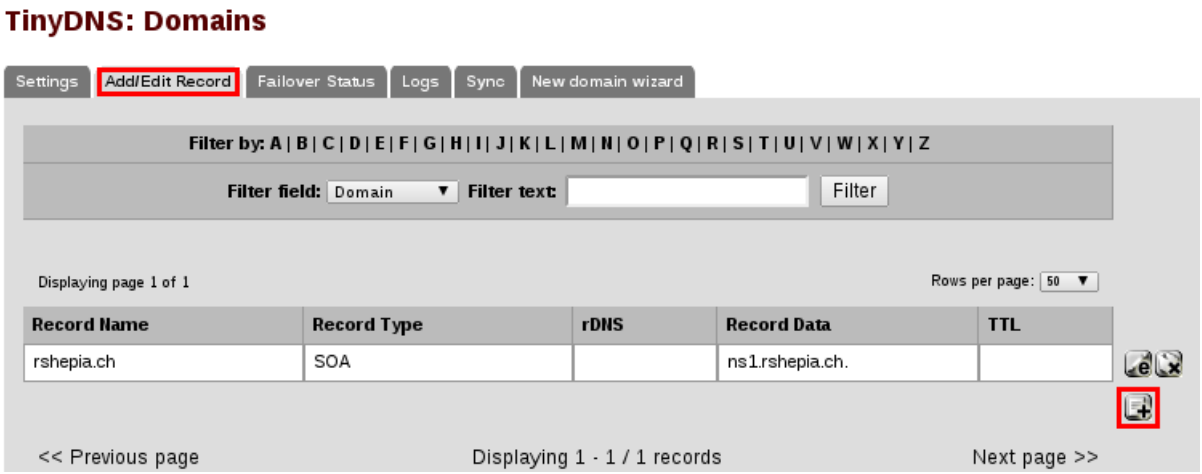



FIG. 4.8 – Création d'un enregistrement

– Avec les paramètres suivants

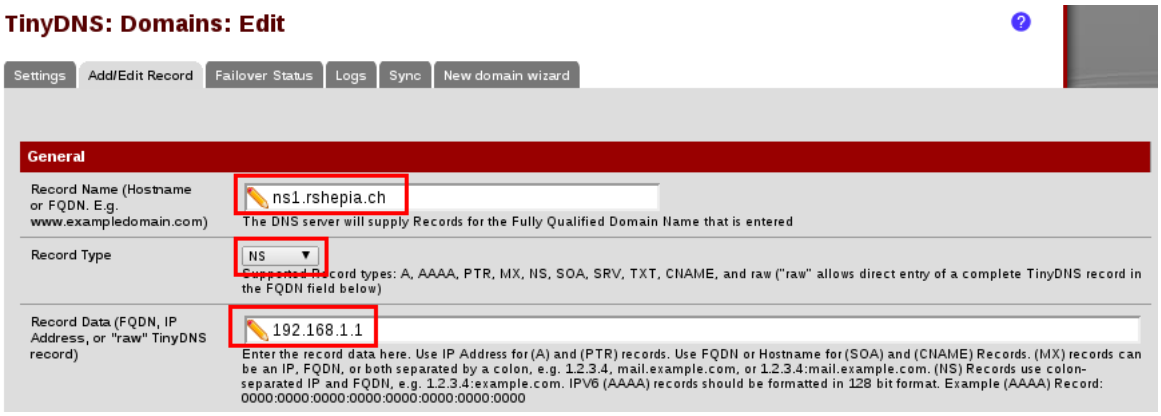



Fig. 4.9 – Paramètres de l'enregistrement NS

– Maintenant que les bases du serveur sont prêtes nous pouvons créer les autres enregistrements. L'opération est identique à celle que nous venons de faire, il faut simplement adapter les champs *Record Name*, *Record Type* et *Record Data* comme ceci

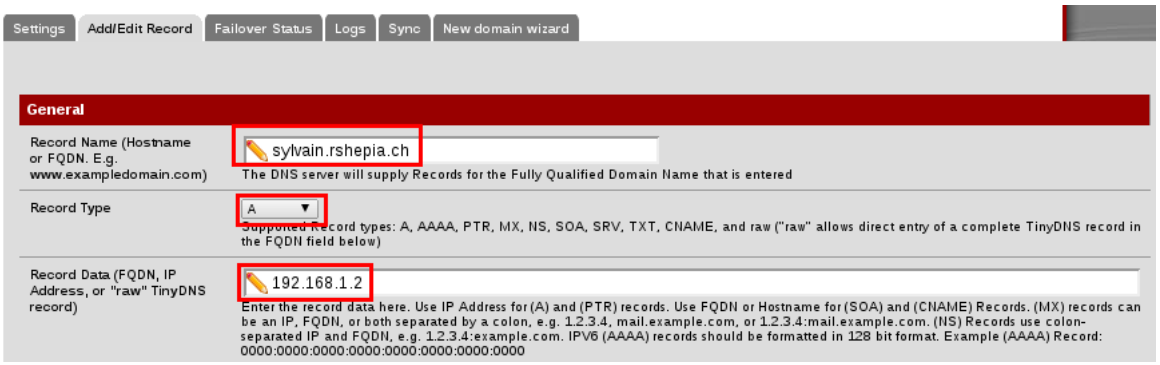



Fig. 4.10 – Paramètres d'un enregistrement A

– Réitéré cette dernière opération pour ajouter l'enregistrement *admin.rshepia.ch*, voir le tableau [Tab. 4.11] pour les détails.

– Une fois tous les enregistrements ajoutés, on doit obtenir cette écran :

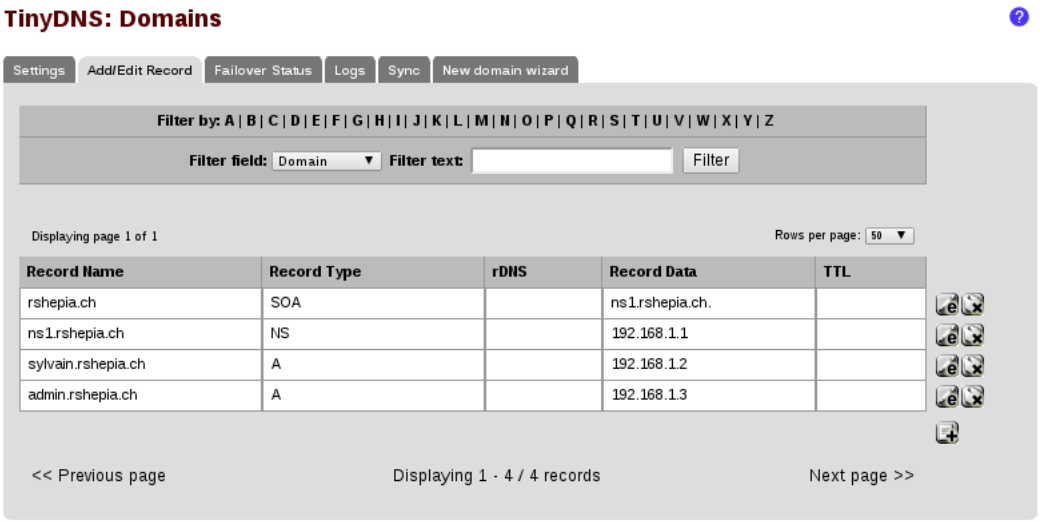



Fig. 4.11 – Enregistrements dans TinyDNS

### 4.5.2 Configuration du fichier *config.xml*

Nous pouvons accéder au Shell depuis la console du serveur en choisissant l'option *8)* ou depuis une connexion SSH si cette dernière est active, ce qui n'est pas le cas par défaut.

Une fois que le paquet TinyDNS a est installé, le fichier de configuration général de PfSense, *config.xml* contient les balises *<installedpackages><tinydns>* et *</tinydns></installedpackages>*.

Pour modifier ce fichier utilisé la commande *vi* :

```
vi /cf/config/config.xml
```

Petit rappel concernant l'utilisation de *vi*. Cette éditeur de texte fonctionne en deux mode : *Commande* ou *Insertion*. Par défaut le programme s'ouvre en mode *Commande*. Pour écrire il faut utiliser la touche *[i]* du clavier pour passé en mode *Insertion*. Pour revenir en mode *Commande* à tous moment, il faut presser la touche *[ESC]*. Pour quitter et enregistrer le document, il faut être en mode *Commande* et saisir *:wq!* puis valider avec la touche *[ENTER]*.

Pour configurer l'adresse IP modifier le contenu entre les balises présentée ci-dessus (ligne 403[3]) :

```
<config>
    <ipaddress>192.168.1.1</ipaddress>
    <enableforwarding/>
    <interface/>
    <dnscache\_listen/>
    <regdhcpstatic/>
    <regdhcp/>
    <enableipmonitoring/>
    <refreshinterval/>
    <syncxmlrpc/>
</config>
```

Aucun domaine et aucun enregistrement sont configurés pour le moment. Nous allons les ajouter entre les balises *<tinydnsdomains>* et *</inydnsdomains>* (ligne 437) qui ne contient rien pour le moment, nous allons y coller la configuration suivante :

[3] utilisé la commande *:set nu* pour afficher les lignes sous vi

```
<tinydnsdomains>
  <config>
    <hostname>rshepia.ch</hostname>
    <recordtype>SOA</recordtype>
    <ipaddress>ns1.rshepia.ch.</ipaddress>
  </config>
  <config>
    <hostname>ns1.rshepia.ch</hostname>
    <recordtype>NS</recordtype>
    <ipaddress>192.168.1.1</ipaddress>
    <ttl/>
    <dist/>
    <srv_port/>
    <srv_priority/>
    <srv_weight/>
    <srv_timestamp/>
    <rdns/>
    <monitorip/>
    <threshhold/>
  </config>
  <config>
    <hostname>sylvain.rshepia.ch</hostname>
    <recordtype>A</recordtype>
    <ipaddress>192.168.1.1</ipaddress>
    <ttl/>
    <dist/>
    <srv_port/>
    <srv_priority/>
    <srv_weight/>
    <srv_timestamp/>
    <rdns/>
    <monitorip/>
    <threshhold/>
  </config>
  <config>
    <hostname>admin.rshepia.ch</hostname>
    <recordtype>A</recordtype>
    <ipaddress>192.168.1.3</ipaddress>
    <ttl/>
    <dist/>
    <srv_port/>
    <srv_priority/>
    <srv_weight/>
    <srv_timestamp/>
    <rdns/>
    <monitorip/>
    <threshhold/>
  </config>
</tinydnsdomains>
```

A présent la configuration TinyDNS est complète. Enregistrer le fichier *config.xml* fraichement modifier et quitter *vi*.

Afin que cette nouvelle configuration soit chargée il faut supprimer le cache contenant l'ancienne configuration avec cette commande

```
rm /tmp/config.cache
```

On peut contrôlé que le fichier de configuration a bien été rechargé en consultant l'interface web, voir §4.11.

## 4.6 Validation des enregistrements DNS dans TinyDNS

Nous pouvons constater que les scripts de configuration ont bien fonctionner en vérifiant le contenu du fichier */var/etc/tinydns/root/data* contient les enregistrement au format TinyDNS à l'aide de la commande *cat*.

```
cat \var\etc\tinydns\root\data

Zrshepia.ch:ns1.rshepia.ch
&rshepia.ch::ns1.rshepia.ch
+sylvain.rshepia.ch:192.168.1.2
+admin.rshepia.ch:192.168.1.3
```

## 4.7 Ouverture du port UDP 53 sur l'interface WAN

Dans la configuration actuelle les requêtes n'aboutiront pas car le Firewall bloque les paquets sur l'interface WAN. Il faut donc ouvrir le Firewall pour autoriser le dialogue avec le service DNS sur le port UDP 53.

– Ouvrir la page de configuration du FireWall depuis le menu *Firewall* > *Rules* puis ajouter la règle suivante :

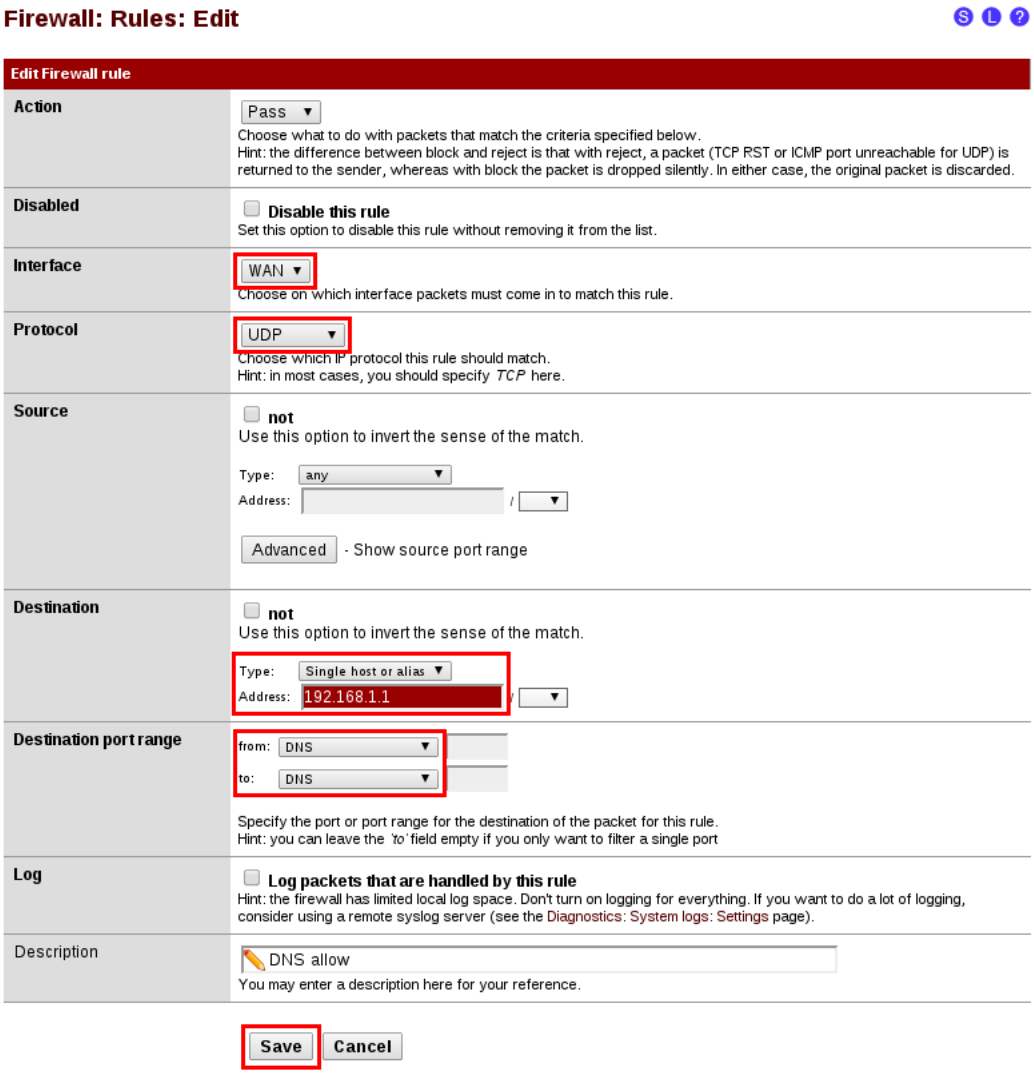



Fig. 4.12 – Régle Firewall autorisant les requêtes DNS

– Cette règle doit apparaitre comme ceci :

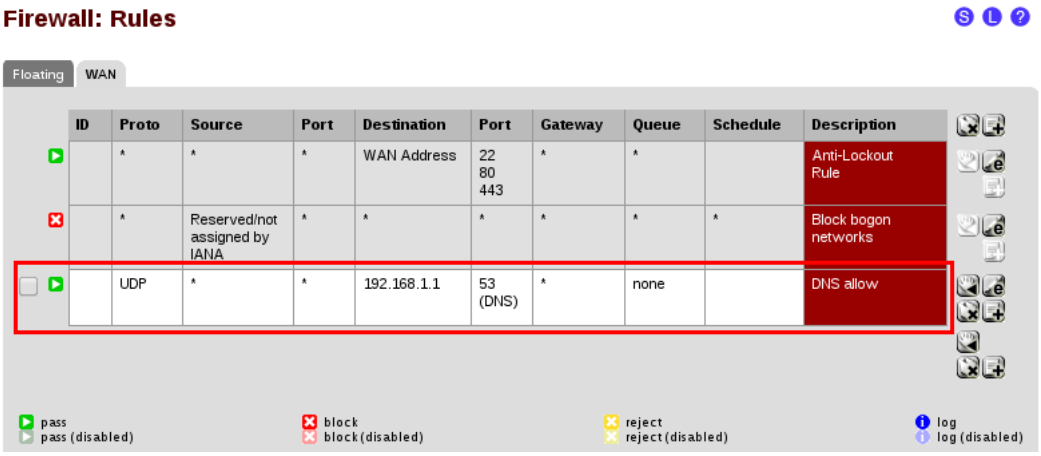



FIG. 4.13 – Régles firewall pour l'interface WAN

# 5 Tests

Une fois notre serveur DNS en place il nous faut pouvoir le tester pour savoir si notre configuration est opérationnel. Plusieurs outils s'offre à nous en fonction de ce que l'on recherche.

## 5.1 Tests de fonctionnement

Le premier outil qui nous vient à l'esprit pour tester le bon fonctionnement d'un serveur DNS est *nslookup* mais il faut savoir qu'il est déprécié et n'est plus supporté dans le monde GNU/Linux et est remplacé par *dig* ou *whois*.

Pour ce travail j'ai décidé d'utilisé *nslookup* car il est est natif sur la plateforme cliente et qu'il est suffisant pour effectuer des tests de fonctionnements.

### 5.1.1 Basique

*Nslookup* peut être utilisé de deux manières, soit en mode interactif, soit en mode non interactif (recherche simple).

Tous les tests qui suivent vont êtres exécutés en mode interactive.

Commençons par ouvrir nslookup comme ceci :

```
C:\Users\albert>nslookup -d2 -domain=rshepia.ch
------------
SendRequest(), len 42
    HEADER:
        opcode = QUERY, id = 1, rcode = NOERROR
        header flags:  query, want recursion
        questions = 1,  answers = 0,  authority records = 0,  additional = 0

    QUESTIONS:
        1.1.168.192.in-addr.arpa, type = PTR, class = IN
------------
DNS request timed out.
    timeout was 2 seconds.
Default Server:  UnKnown
Address:  192.168.1.1
>
```

L'option *-d2* active le debugeur du deuxième niveau et l'option *-domain=rshepia.ch* défini le domaine utilisé par défaut pour les requêtes DNS. J'ai activé le mode de debugage afin de visualiser les requêtes émise pas l'outil et de les corréler avec la capture des parquets réseau.

Nous n'avons fait que ouvrir *nslookup* mais l'on peut constater qu'une requête a déjà été émise. Par défaut cette outil fait une requête de type PTR (inverse) sur l'adresse du serveur DNS définis dans les paramètres réseau de Windows.

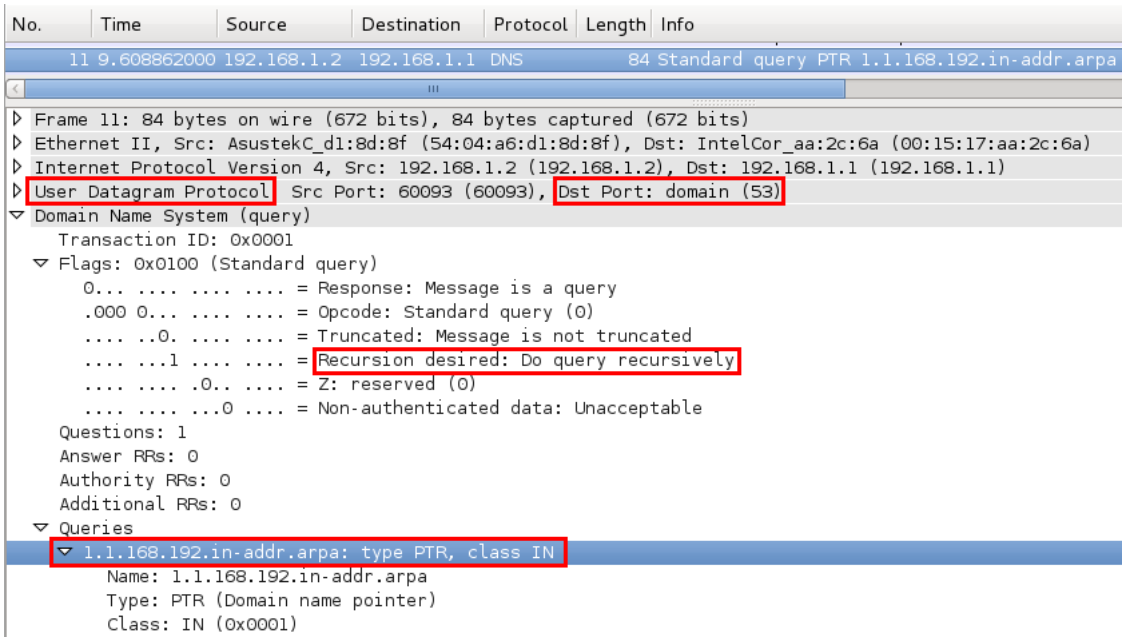



Fig. 5.1 – Requête DNS : PTR sur serveur DNS

On peut constater que l'on retrouve les mêmes informations dans le debug et dans la capture. Il n'y a pas de réponse car le serveur DNS ne fait pas de résolution inverse.

Maintenant nous pouvons faire le test suivant :

```
> admin
Server:  UnKnown
Address:  192.168.1.1
------------
SendRequest(), len 34
    HEADER:
        opcode = QUERY, id = 8, rcode = NOERROR
        header flags:  query, want recursion
        questions = 1,  answers = 0,  authority records = 0,  additional = 0
    QUESTIONS:
        admin.rshepia.ch, type = A, class = IN
------------
------------
Got answer (68 bytes):
    HEADER:
        opcode = QUERY, id = 8, rcode = NOERROR
        header flags:  response, auth. answer, want recursion
        questions = 1,  answers = 1,  authority records = 1,  additional = 0
    QUESTIONS:
        admin.rshepia.ch, type = A, class = IN
    ANSWERS:
    ->  admin.rshepia.ch
        type = A, class = IN, dlen = 4
        internet address = 192.168.1.3
        ttl = 86400 (1 day)
    AUTHORITY RECORDS:
    ->  rshepia.ch
        type = NS, class = IN, dlen = 6
        nameserver = ns1.rshepia.ch
        ttl = 259200 (3 days)
------------
------------
SendRequest(), len 34
    HEADER:
        opcode = QUERY, id = 9, rcode = NOERROR
        header flags:  query, want recursion
        questions = 1,  answers = 0,  authority records = 0,  additional = 0
    QUESTIONS:
        admin.rshepia.ch, type = AAAA, class = IN
------------
------------
Got answer (85 bytes):
    HEADER:
        opcode = QUERY, id = 9, rcode = NOERROR
        header flags:  response, auth. answer, want recursion
        questions = 1,  answers = 0,  authority records = 1,  additional = 0
    QUESTIONS:
        admin.rshepia.ch, type = AAAA, class = IN
    AUTHORITY RECORDS:
    ->  rshepia.ch
        type = SOA, class = IN, dlen = 39
        ttl = 2560 (42 mins 40 secs)
        primary name server = ns1.rshepia.ch
        responsible mail addr = hostmaster.rshepia.ch
        serial  = 1359045794
        refresh = 16384 (4 hours 33 mins 4 secs)
        retry   = 2048 (34 mins 8 secs)
        expire  = 1048576 (12 days 3 hours 16 mins 16 secs)
        default TTL = 2560 (42 mins 40 secs)
------------
Name:    admin.rshepia.ch
Address:  192.168.1.3
>
```

Nous retrouvons les quatre paquets au niveau de cette capture :

| No. | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|
| 58 | 192.168.1.2 | 192.168.1.1 | DNS | 76 | Standard query A admin.rshepia.ch |
| 59 | 192.168.1.1 | 192.168.1.2 | DNS | 110 | Standard query response A 192.168.1.3 |
| 60 | 192.168.1.2 | 192.168.1.1 | DNS | 76 | Standard query AAAA admin.rshepia.ch |
| 61 | 192.168.1.1 | 192.168.1.2 | DNS | 127 | Standard query response |

FIG. 5.2 – Requête DNS : admin.rshepia.ch

Le premier constat est que l'on peut faire est que *nslookup* pose deux questions au serveur DNS, la première est de type *A* et la seconde est de type *AAAA*. Par défaut *nslookup* recherche le nom d'un hôte en IP *version 4* et *version 6*.

Il est possible de changé le type pour le quelque on veut le FQDN au travers de l'option *set type=*. On pourrait utiliser le type *ANY* qui nous retournerait tous les type d'enregistrement disponible pour un FQDN, cette option n'est pas utile dans notre cas car elle nous retournerai le même résultat mais peut être utilisé dans le cadre de la configuration d'un serveur mail ou le FQDN *mail.monsite.ch* pourrait avoir un enregistrement de type *MX* et *A*.

Le test que nous venons d'effectuer nous prouve que le serveur DNS résout nos requêtes.

### 5.1.2 Enregistrement inexistant

On reconnait une réponse négative de deux manière :

- Le mode debug de *nslookup* ne nous affiche pas la partie *ANSWERS* dans la réponse *Got answer*. comparé la réponse de type *A* et *AAAA* [5.1.1]
- Le paquet de réponse comporte le Flags suivant :

```
Flags: 0x8503 (Standard query response, No such name)
    1... .... .... .... = Response: Message is a response
    .000 0... .... .... = Opcode: Standard query (0)
    .... .1.. .... .... = Authoritative: Server is an authority for domain
    .... ..0. .... .... = Truncated: Message is not truncated
    .... ...1 .... .... = Recursion desired: Do query recursively
    .... .... 0... .... = Recursion available: Server can't do recursive queries
    .... .... .0.. .... = Z: reserved (0)
    .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
    .... .... ...0 .... = Non-authenticated data: Unacceptable
    .... .... .... 0011 = Reply code: No such name (3)
```

FIG. 5.3 – FLAG en cas de réponse négative

Si on obtient aucun paquet en retour après une requête cela veut dire que le serveur n'est pas disponible, soit parce que le service n'est pas actif, soit parce qu'il y a un problème au niveau du réseau (FireWall, collision, timeout, etc).

### 5.1.3 Changement d'un enregistrement DNS

Maintenant qu'on sait que notre serveur réponds, il faut savoir si les changement sont bien pris en compte.

Dans un premier temps, il faut modifier l'adresse ip d'un des enregistrement sur le serveur.

Deuxième, il faut vider le cache DNS se trouvant sur le client qui est automatiquement mis à jour lors de la réception d'une réponse à une requête DNS, à l'aide de la commande suivante :

```
ipconfig /flushdns
```

Troisièmement on peut faire un test plus simple que le précédent grâce à l'outil *ping* qui va automatiquement générer la requête DNS. Au préalable, il m'a fallu changer l'adresse ip de mon ordinateur portable afin qu'il réponde au requête ICMP.

```
C:\Users\albert>ping admin.rshepia.ch

Pinging sylvain.rshepia.ch [192.168.1.5] with 32 bytes of data:
Reply from 192.168.1.5: bytes=32 time<1ms TTL=128
Reply from 192.168.1.5: bytes=32 time<1ms TTL=128
Reply from 192.168.1.5: bytes=32 time<1ms TTL=128
Reply from 192.168.1.5: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.5:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
```

### 5.1.4 Requête TCP

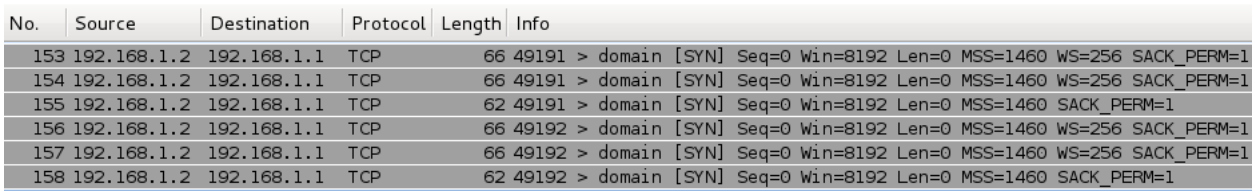
Toutes les paquets présentés dans ce chapitre sont transportés par le protocole UDP, ce qui nous a permit de montré que *TinyDNS* utilise bien ce protocole. Le test qui suit va nous démontrer qu'il n'utilise pas le protocole TCP.

En l'état le FireWall est configuré pour ne laisser passé que le protocole UDP sur port 53, j'ai du adapté la règle pour ouvrir le protocole TCP sur le même port afin de réaliser ce test.

Nslookup permet de forcer les requêtes sur le port 53 du protocole TCP avec l'option *vc*.

```
> set vc
> sylvain
Server:  UnKnown
Address:  192.168.1.1
------------
SendRequest(), len 36
    HEADER:
        opcode = QUERY, id = 17, rcode = NOERROR
        header flags:  query, want recursion
        questions = 1,  answers = 0,  authority records = 0,  additional = 0
    QUESTIONS:
        sylvain.rshepia.ch, type = A, class = IN
------------
connect failed: Result too large
SendRequest failed
------------
SendRequest(), len 36
    HEADER:
        opcode = QUERY, id = 18, rcode = NOERROR
        header flags:  query, want recursion
        questions = 1,  answers = 0,  authority records = 0,  additional = 0
    QUESTIONS:
        sylvain.rshepia.ch, type = AAAA, class = IN
------------
connect failed: Result too large
SendRequest failed
*** UnKnown can't find sylvain: Unspecified error
>
```

La fin du debug nous stipule une erreur mais il nous fourni pas plus de précision. Une capture Wireshark nous en dira plus.



| No. | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|
| 153 | 192.168.1.2 | 192.168.1.1 | TCP | 66 | 49191 > domain [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 154 | 192.168.1.2 | 192.168.1.1 | TCP | 66 | 49191 > domain [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 155 | 192.168.1.2 | 192.168.1.1 | TCP | 62 | 49191 > domain [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1 |
| 156 | 192.168.1.2 | 192.168.1.1 | TCP | 66 | 49192 > domain [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 157 | 192.168.1.2 | 192.168.1.1 | TCP | 66 | 49192 > domain [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 158 | 192.168.1.2 | 192.168.1.1 | TCP | 62 | 49192 > domain [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1 |

Fig. 5.4 – Requête DNS sur le protocole TCP

On constate très clairement que les paquets TCP vont du client au serveur et qu'il n'y a aucun retour, c'est qu'il n'y a aucun service qui répond sur le serveur. Les requêtes TCP sont donc ignorée par *TinyDNS*, ce qui est le role de *axfrdns* qui n'est pas activé.

## 5.2 Tests de sécurités

Je n'ai pas trouvé de test de sécurité simple à la hauteur de *TinyDNS* car il est naturellement protégé contre les attaques les plus courantes. Les points faible d'un serveur DNS sont le cache (cache-poisonning), la récursivité, la délégation d'autorité, hors TinyDNS ne s'occupe de répondre que aux requêtes non récursive. Le seul point faible qui je vois, serait une attaque par déni de service (DDOS) mais il n'y a pas beaucoup d'application qui arrive à y faire face nativement. Cette attaque est dur à simulé avec le matériel utilisé dans ce laboratoire.

Ce n'est pas parce que TinyDNS est très robuste qu'il n'est pas possible de le sécurisé encore plus. Depuis les attaques de 2008, l'IETF, à créé le protocole DNSSEC[1] (*Domain Name System Security Extensions*) pour palier aux problèmes de sécurité du protocole DNS.

On peut rendre TinyDNS compatible avec DNSSEC grace au site http://www.tinydnssec.org/. Je n'ai trouvé aucune information mentionnant que le paquet pour PfSense implémentait ce protocole.

[1] normalisé par le RFC 4033[4]

# 6 Difficultés rencontrées

Le site du créateur de TinyDNS n'est pas rédigé dans un anglais facilement accessible, cela ma pris beaucoup de temps pour comprendre les tournures de phrases et les mécanismes utilisés.

J'ai commencé mon projet à l'envers, j'ai débuté par la mise en place du laboratoire et la configuration de ce dernier au lieu d'effectuer les recherches et de définir les étapes qui structures mon travail. Cette mauvaise approche m'a fait prendre conscience, trop tard, de mon incompréhension du cahier des charges ce qui m'a pénalisé en terme de temps.

Venant du monde professionnel, j'ai interpréter l'objectif final *changement du serveur DNS du domaine tdeig.ch* comme une migration, je suis donc partie sur la planification d'un plan de migration, alors qu'il s'agissait plutôt d'une évaluation en terme de sécurité du futur produit afin qu'une migration du serveur DNS soit planifiée sur la base de mon travail.

La configuration ne fonctionnait pas tous de suite à cause du Firewall activé par défaut. Il m'a fallu ajouter une règle pour autoriser le protocole UDP sur le port 53.

La sécurité autour de service DNS est très complexe et il est difficile de l'abordé sans être un spécialiste de ce protocole.

Des problèmes d'organisation et technique (ordinateur portable en train de tomber en panne) mon retardés dans mes recherches et la rédaction de mon travail que je ne pouvais pas toujours réalisé à l'école.

Le HUB n'étant pas de toute jeunesse, des ports ne fonctionnent pas et il se déconnecte de manière aléatoire (port utilisés 12, 8 et 4 du bloc A sur le module gauche du HUB)

# 7 Conclusion technique

Il est relativement simple d'implémenter TinyDNS au seins du système d'exploitation PfSense. On a tous de même besoins d'une bonne connaissance de PfSense et de TinyDNS si l'on désire réaliser une configuration sans l'aide de la console d'administration web.

Ce travail démontre que TinyDNS peut avoir sa place sur un Firewall sans pour autant compromettre la sécurité de ce dernier. Il sera facile de faire une migration d'un serveur BIND vers ce serveur DNS simplement en recréant les enregistrements.

On peut constater que TinyDNS est nativement protégé et peut répondre facilement à une haute exigence en matière de sécurité. Il faut cependant une bonne connaissance du DNS afin de bien configuré ce serveur car l'information est plus dificile à trouver que pour BIND.

# 8 Conclusions Personnelles

Dans un premier temps j'ai eu de la penne à me concentrer sur le sujet lui même car je me suis concentré sur le document que j'ai rédigé en Latex. Pourquoi ai-je fais se choix alors que rien n'a été demandé ou imposé ? J'ai toujours préférer utiliser les bons outils pour travailler. Je me suis tourné vers Latex car il a été inventé pour créé des articles scientifiques ou des mémoires. Ce travail est considéré comme un mémoire donc l'outil me paraissait idéal. L'orientation de l'école à ce tourné vers des produits OpenSource ma motivé à faire pas vers cette outils de rédaction. Je ne pense pas avoir perdu plus de temps qu'avec une autre solution (sans compter l'apprentissage, qui est relativement rapide) car il est répartie autrement. La mise en page ne se fait pas (ou très peu) à la fin, comme nous pousse à faire d'autres programmes, mais au début en définissant notre modèle et en cours de rédaction.

Ce travail a été décousu dans son ensemble. Des problèmes d'ordre privé et technique on en piété sur le temps apartie en début de semestre et il n'a pas été facile de rattrapé le temps perdu. Quelques problèmes de communications concernant le cahier des charges sont aussi venu pertubé ce travail.

Malgré ces différents événements, j'ai tous mis en oeuvre pour porté ce travail à terme. L'énoncé me semblait simple mais j'ai buté sur beaucoup de problèmes au niveau de l'intégration de TinyDNS dans PfSense car il n'y pas de documentation sur ce paquage. J'ai d'abord concentré mon energie sur TinyDNS en oubliant complétement son intégration dans PfSense qui impose un mécanise de configuration complétement différent.

# A Annexes

## A.1 Historique

Évolution d'une architecture Nagios mono serveur en une architecture redondante multi-serveur au saint de l'entreprise.

L'extension de l'architecture de notre outil de supervision a été écarté pour ce projet car il ne fait pas l'objet d'un travail d'étude. Ce sujet pourra peut être retenu pour le travail de Bachelor, le cahier des charges est à définir au saint de l'entreprise et avec l'encadrement d'un professeur.

# Lien & références


[1] Annonce de la récompence. http ://cr.yp.to/djbdns/guarantee.html.

[2] Bug trouvé dans djbdns/axfrdns. http ://article.gmane.org/gmane.network.djbdns/13864.

[3] Comparaison des serveurs dns. http ://en.wikipedia.org/wiki/Comparison_of_DNS_server_software.

[4] Dnssec, rfc 4033. http ://tools.ietf.org/html/rfc4033.

[5] Liste des serveurs dns root. http ://www.root-servers.org.

[6] Pfsense. http ://www.pfsense.org/.

[7] Philosophie d'unix. http ://fr.wikipedia.org/wiki/Philosophie_d'Unix.

[8] Structure du fichier data. http ://cr.yp.to/djbdns/tinydns-data.html.

[9] CERT vulnérabilité vu#800113. http ://www.kb.cert.org/CERT_WEB%5Cservices%5Cvulnotes.nsf/id/800113.

[10] Tinydns. http ://tinydns.org/.