Réseau et services pour la plateforme informatique

IMA 4 SC

CONG CHEN

Remerciement

Ce stage, je tiens à remercier à Monsieur Xavier Redon, mon encadrant du stage, qui m'a beaucoup aidé pendant mon stage.

Je remercie aussi à Monsieur Yizhou Lin et Madame Hongyu Zhang, qui m'ont beaucoup aidé avec l'avancement de mon stage.

Abstract

Ce rapport présente le stage que j'ai réalisé dans le cadre de ma seconde année à l'école d'ingénieurs Polytech'Lille dans la spécialité IMA. Le stage s'est déroulé au sein de la plateforme pédagogique informatique de l'école de mi-mai à fin juin. Le stage fut pour moi l'occasion de découvrir le monde des réseaux informatiques et particulièrement du système de nommage DNS. L'objectif de la plateforme est de réaliser un réseau informatique distinct de celui de l'école pour les expérimentations pédagogiques pour éviter de perturber le réseau de production lors des travaux pratiques. Le stage s'est concentré sur le système de nommage DNS d'Internet et sa version sécurisé DNSSEC. Le rapport contient aussi un manuel technique concernant la configuration d'un serveur DNS.

This report is aimed to present my internship carried out in Polytech' Lille. This internship was carried out at the end of my first Master year of System Communication in Polytech' Lille, from the middle of May to the end of June. This is an occasion for me to explore the world of network especially in the domain of DNS and to develop my experiences acquired from my study. This internship was achieved with the help of M Xavier Redon, having a goal of creating the informatic network for the informatic platform of Polytech' Lille firstly and of being familiar with the experiences in Second Master year. During the internship, it is necessary to have acknowledges of DNS, DNSSEC and DNSSEC-Tools. This report also contains the commands that are applied to achieve the securisation of seveur of the platform.

1 Introduction

1.1 Contexte

1.1.1 *Introduction du sujet <Réseau et service pour la plateforme informatique >*

La principale tâche réalisée durant le stage est la création d'un serveur de nommage DNS sécurisé (DNSSEC) pour la plateforme pédagogique informatique. La création se fait en utilisant des outils standards (DNSSEC-TOOLS) mais aussi en développant un système de Makefile associé à quelques utilitaires écrits en C.

1.2 Objectif du stage

L'objectif du stage est d'abord de créer un réseau informatique pour la plateforme de Polytech' Lille. Pour l'instant la plateforme utilise le réseau de production. Il convient d'isoler les deux réseaux pour éviter un impact négatif lors des manipulations des élèves.

2 La manipulation

2.1 Machine Virtuelle

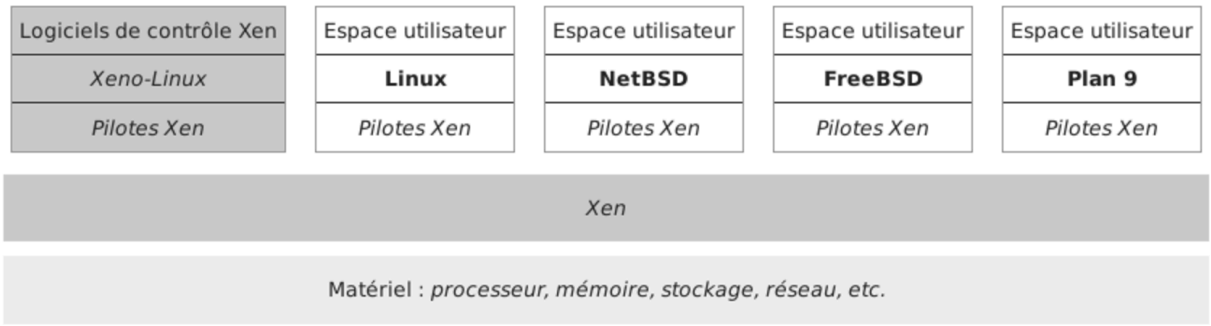
2.1.1 *Introduction de Xen*

Le serveur DNSSEC est installé dans une machine virtuelle utilisant la technologie Xen.

Xen est un logiciel libre de virtualisation, plus précisément un hyperviseur de machine virtuelle. Xen permet d'exécuter plusieurs systèmes d'exploitation (et leurs applications) de manière isolée sur une même machine physique sur plate-forme x86, x86-64, IA-64 et PowerPC, ARM Cortex-A7 et Cortex-A151 (bientôt sur SPARC). Les systèmes d'exploitation invités partagent ainsi les ressources de la machine hôte.

2.1.2 Architecture de Xen

Chaque système d'exploitation invité tourne dans un « domaine ». Xen est une fine couche fonctionnant directement sur le matériel.



2.1.3 Réalisation d'une machine virtuel à l'aide de Xen

Plutôt que de travailler directement sur le serveur DNS déjà en chantier, j'ai créé ma propre machine virtuelle. La création s'est fait en utilisant le script xen-create-image. …

Pour réaliser la création de ma machine virtuelle, j'utilise la commande ci-dessous :

xen-create-image --hostname=cong – dns --ip=5.23.44.83 --netmask=255.255.255.248 --dist=jessie --gateway=5.23.44.81 --dir=/usr/local

2.2 DNS

2.2.1 Introduction de DNS

Une fois la machine virtuelle créée, la configuration DNS peut être envisagée.

Le Domain Name System (ou DNS, système de noms de domaine) est un service permettant de traduire un nom de domaine en informations de plusieurs types qui y sont associées, notamment en adresses IP de la machine portant ce nom.

Le système des noms de domaines consiste en une hiérarchie dont le sommet est appelé la racine. On représente cette dernière par un point. Dans un domaine, on peut créer un ou plusieurs sous-domaines ainsi qu'une délégation pour ceux-ci, c'est-à-dire une indication que les informations relatives à ce sous-domaine sont enregistrées sur un autre serveur. Ces sous-domaines peuvent à leur tour déléguer des sous-domaines vers d'autres serveurs.

Tous les sous-domaines ne sont pas nécessairement délégués. Les délégations créent des zones, c'est-à-dire des ensembles de domaines et leurs sous-domaines non délégués qui sont configurés sur un serveur déterminé. Les zones sont souvent confondues avec les domaines.

Les domaines se trouvant immédiatement sous la racine sont appelés domaine de premier niveau (TLD : Top Level Domain). Les noms de domaines ne correspondant pas à une extension de pays sont appelés des domaines génériques (gTLD), par exemple .org ou .com. S'ils correspondent à des codes de pays (fr, be, ch…), on les appelle ccTLD (country code TLD).

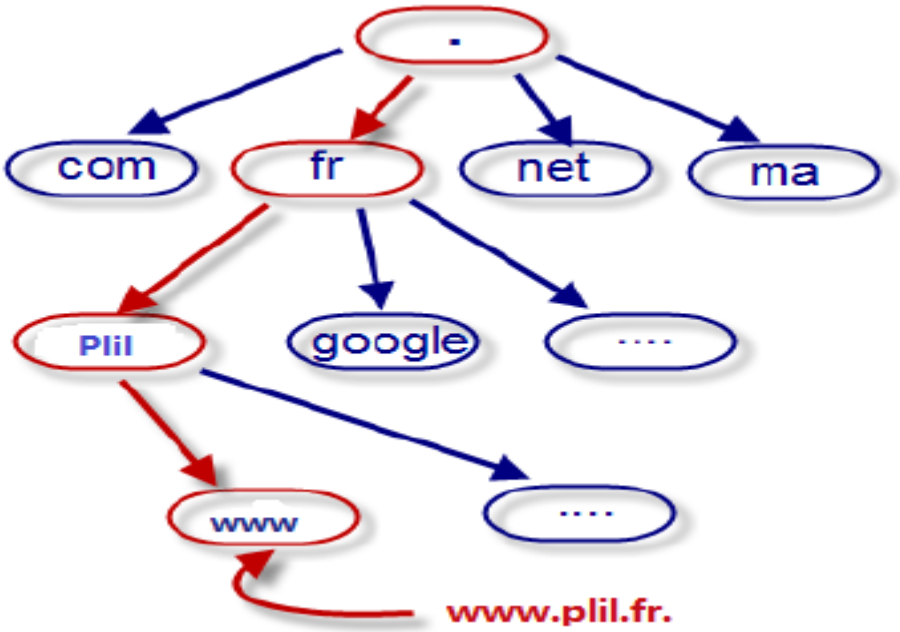
On représente un nom de domaine en indiquant les domaines successifs séparés par un point, les noms de domaines supérieurs se trouvant à droite. Par exemple, le

domaine fr. est un TLD, sous-domaine de la racine. Le domaine plil.fr. est un sous-domaine de .fr. Cette délégation est accomplie en indiquant la liste des serveurs DNS associée au sous-domaine dans le domaine de niveau supérieur.

Les noms de domaines sont donc résolus en parcourant la hiérarchie depuis le sommet et en suivant les délégations successives, c'est-à-dire en parcourant le nom de domaine de droite à gauche.

Pour qu'il fonctionne normalement, un nom de domaine doit avoir fait l'objet d'une délégation correcte dans le domaine de niveau supérieur.

Voice une image qui représente la hiérarchie de DNS



Pendant mon stage, j'utilise un nom de domaine **plil.info** pour associer un nom DNS à mon adresse IP **5.23.44.83**. Après la réservation de ce nom de domaine, j'ai configuré bind (paquetage bind9) sur ma machine virtuelle cong-dns pour donner les adresses correspondant à mon nom de réseau IP, au nom de mon interface de routeur, au nom de ma machine et au nom de l'adresse de diffusion de mon réseau IP.

```
$TTL 259200
@ IN SOA  pill.info. postmaster.plil.info. (
3                ; Version
7200             ; Refresh (2h)
```

```
3600                ; Retry   (1h)
1209600             ; Expire (14j)
259200 )            ; Minimum TTL (3j)
   IN NS ns.plil.info.
   IN NS ns6.plil.info.
www    IN A      5.23.44.83
ns         IN A      5.23.44.83
```

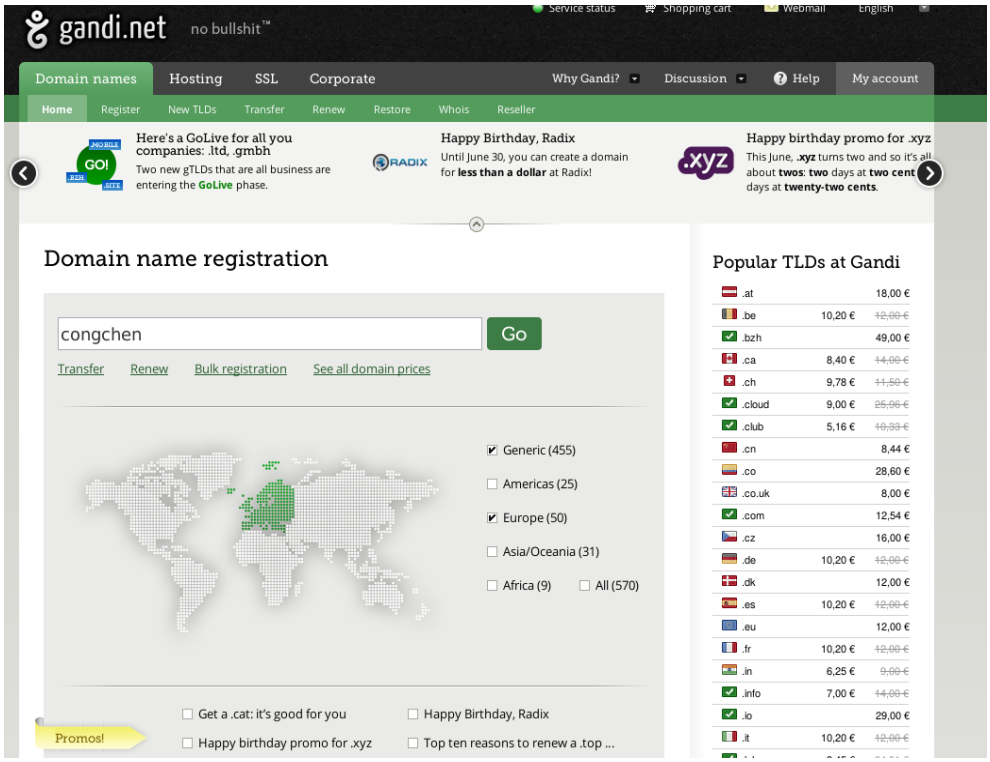
Le bind9 est maître pour la zone plil.info

```
zone "plil.info" {
        type master;
        file "polytech-lille.fr/polytech-lille";
            };
```

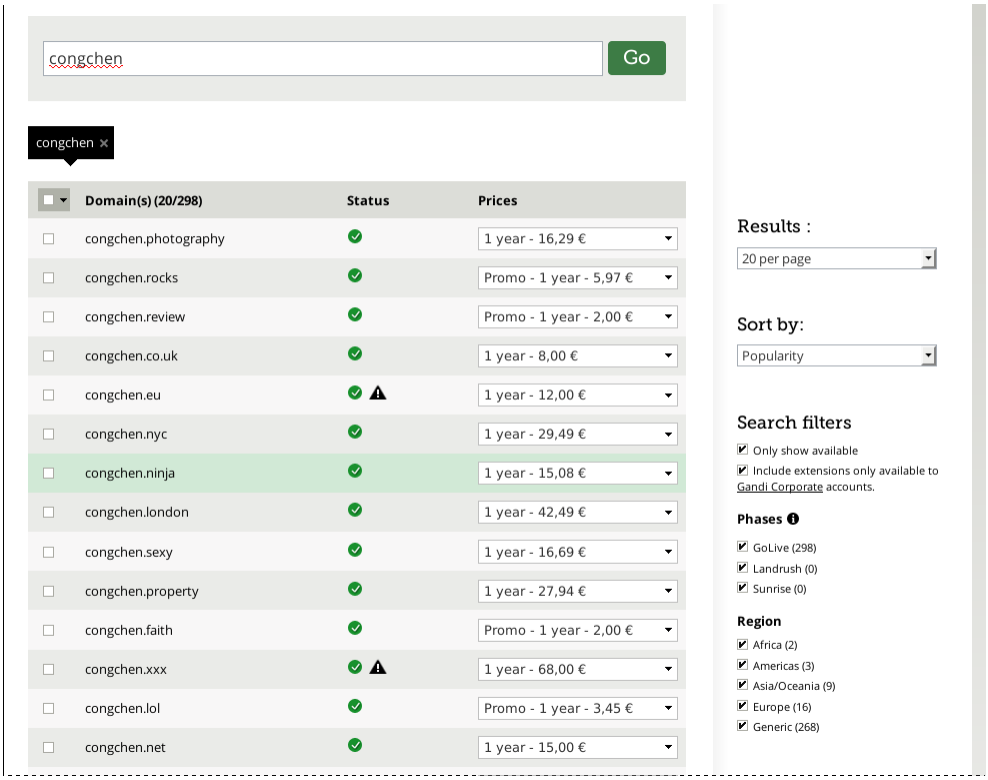
Pour mieux comprendre le fonctionnement de DNS j'ai aussi réservé le domaine congchen.fr.

Pour effectuer cette réservation j'ai choisi le registrar Gandi (gandi.net).

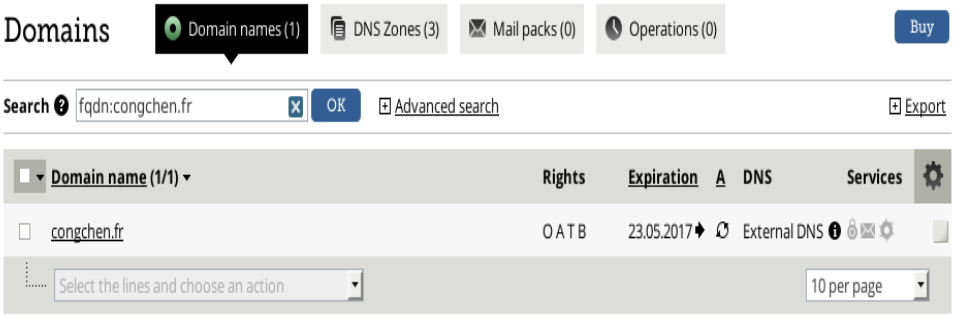
Dans le champ texte « domain name registration », j'ai choisi le nom « congchen »



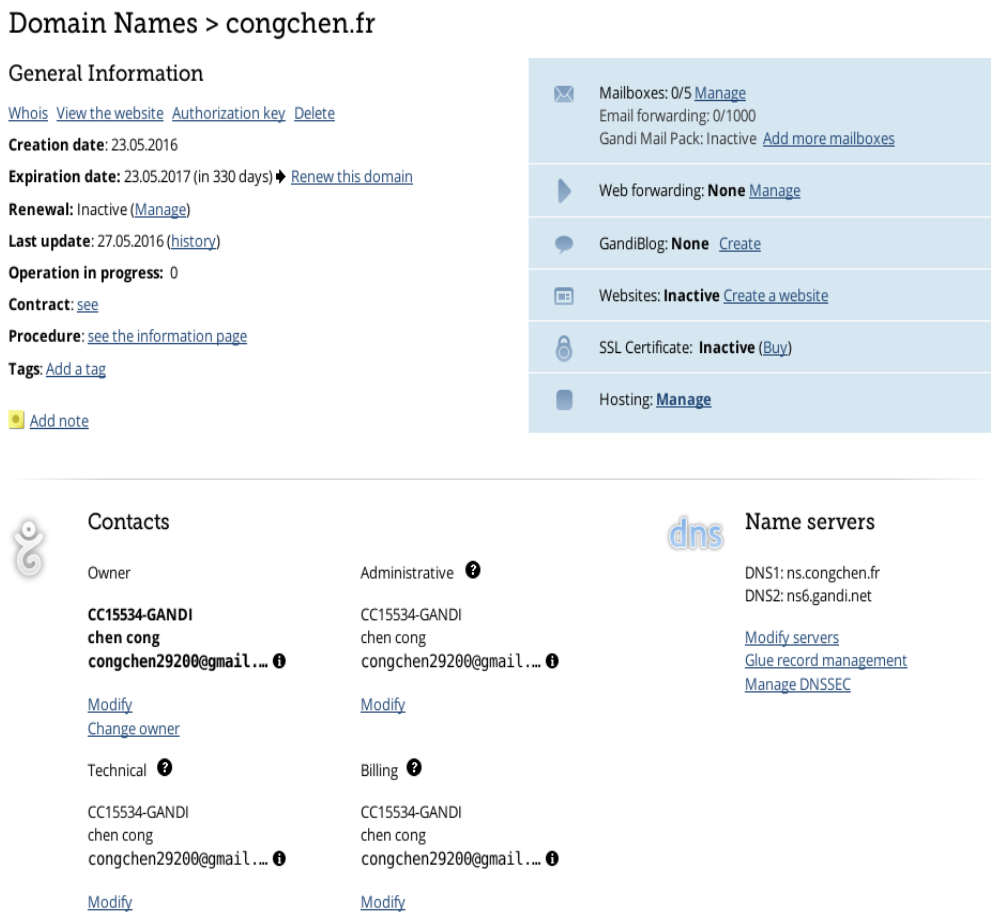
Je pouvais ensuite choisir l'extension .fr



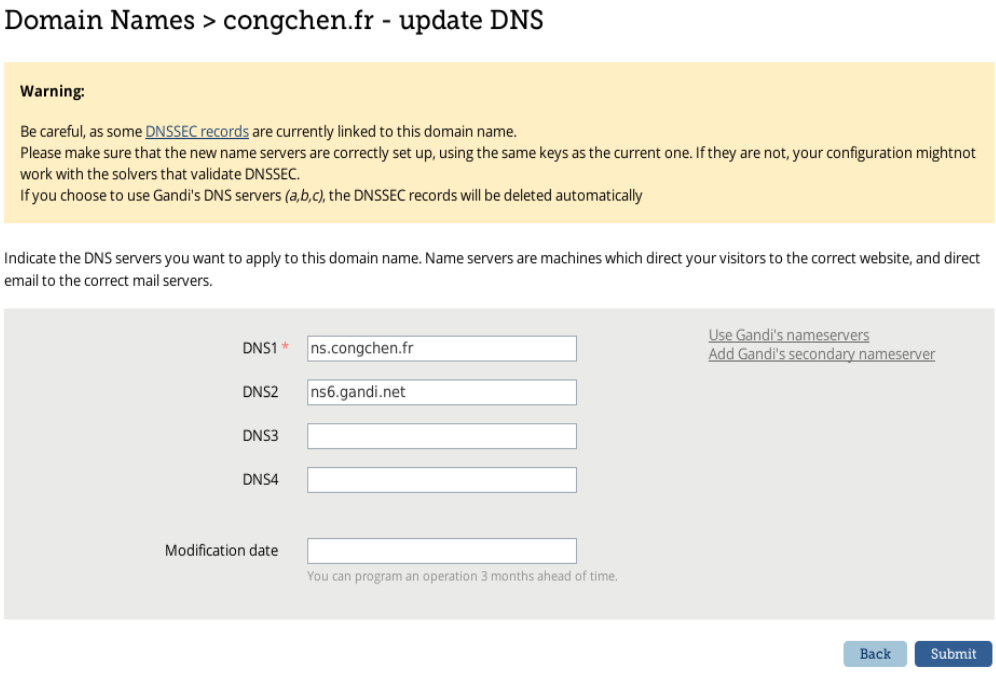
Pour déléguer la gestion du nom de domaine « congchen.fr » vers le serveur DNS de test, il faut sélectionner ce nom de domaine dans l'interface de gandi.



Après il faut cliquer congchen.fr, j'ai obtenu la page suivante :



Sur cette page en bas à droite, il y a un lien pour modifier les serveurs DNS du nom de domaine.



2.3 DNSSEC

2.3.1 Introduction de DNSSEC

Des cyber-attaques visent DNS, c'est pourquoi ce protocole possède une version securisé DNSSEC. Et le protocole DNSSEC existe depuis 2005.

DNSSEC est basé sur un modèle de cryptographie à clé publique afin d'assurer l'authenticité et l'intégrité des réponses. Un serveur d'autorité calcule un hash de la réponse puis le signe avec une clé privée (secrète) avant d'envoyer le paquet et son hash au résolveur. Celui-ci sera à même de vérifier l'authenticité et l'intégrité des données en vérifiant que le hash déchiffré à l'aide de la clé publique associée à la clé privée correspond bien au hash recalculé de la réponse.

2.3.2 Pour sécuriser bind9, une implantation Unix du DNS

1. Il faut ajouter l'option **dnssec-enable yes**; dans le fichier **named.conf.options;**

```
options {
        directory “/var/cache/bind”;
        dnssec-validation auto;
        dnssec-enable yes;
        auth-nxdomain no;
        listen-on-v6 { any; };
        allow-transfer {“allowed_to_transfer”;};
        }
acl “allowed_to_transfer”{
        217.70.177.40/32;
        }
```

2. Il faut créer un répertoire de nom **plil.info.dnssec** pour y générer les clefs ;
3. Il faut créer la clef asymétrique de signature de clefs de zone (pour accélérer la génération sur un système de test nous pouvons utiliser l'option -r

/dev/urandom)

dnssec-keygen –r /dev/urandom –a RSASHA1 –b 2048 –f KSK –a ZONE plil.info

4. Il faut créer la clef asymétrique de la zone pour signer les enregistrements (pour accélérer la génération sur un système de test nous pouvons utiliser l'option -r /dev/urandom);

dnssec-keygen –r /dev/urandom –a RSASHA1 –b 1024 –n ZONE plil.info

5. Pour plus de claret, nous renommons les deux paires de clefs obtenues en utilisant le nom de la zone comme préfixe puis en suffixant d'abord par la destination de la clef (-ksk pour la KSK ou -zsk pour la ZSK) puis par le type de clef (.key pour la clef publique ou private pour la clef privée);

En conséquence, après toutes ces étapes j'ai 5 fichiers dans le répertoire plil.info.dnssec:

dsset-plil.info.
plil.info-ksk.key
plil.info-ksk.private
plil.info-zsk.key
plil.info-zsk.private

6. Il faut inclure les clefs publiques dans le fichier de zone et incrémenter le numéro de version de la zone;

$include /etc/bind/pill.info.dnssec/pill.info-ksk.key
$include /etc/bind/pill.info.dnssec/pill.info-zsk.key

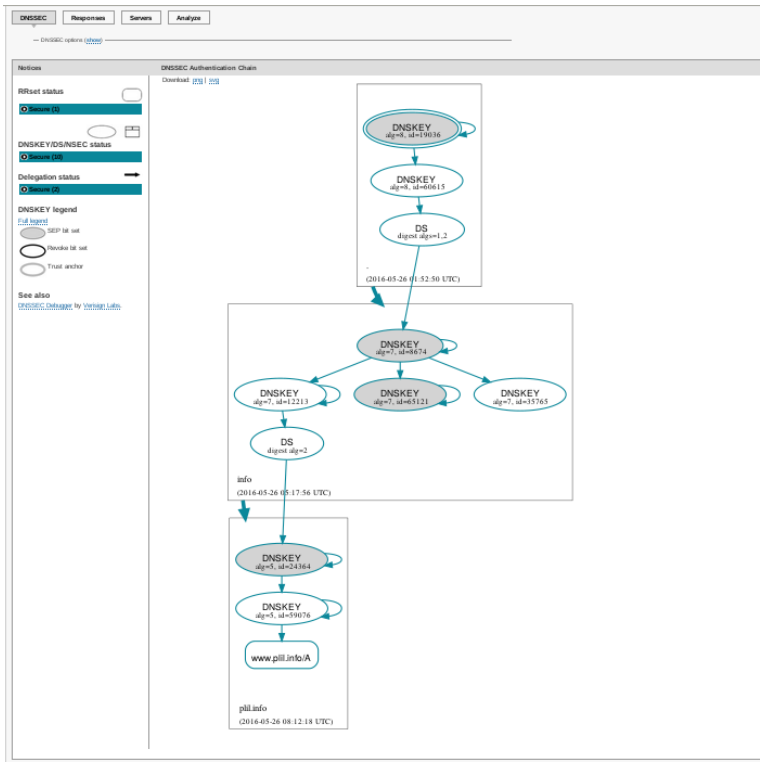
Pour le DNS congchen.fr, j'ai fait les mêmes configurations.

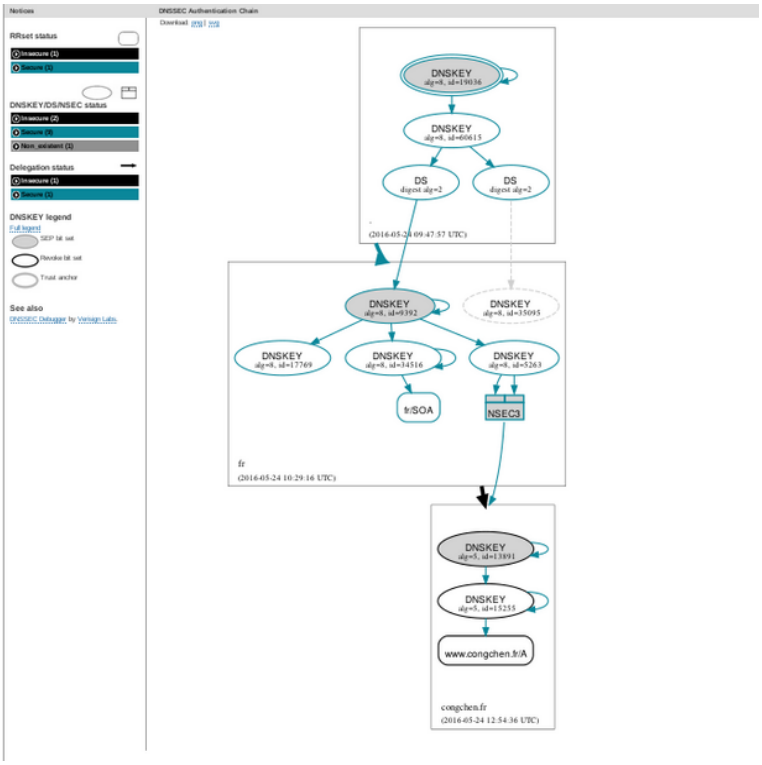
Ensuite il reste encore deux étapes à faire:

1. Il faut aussi modifier le fichier named.conf.local pour utiliser la zone signée de suffixe .signed ;

2. Il faut communiquer la partie publique de la KSK (présente dans le fichier plil.info-ksk.key) au registrar Gandi.

La visualization de la chaîne de certification DNSSEC pour **plil.info** et **congchen.fr** est :

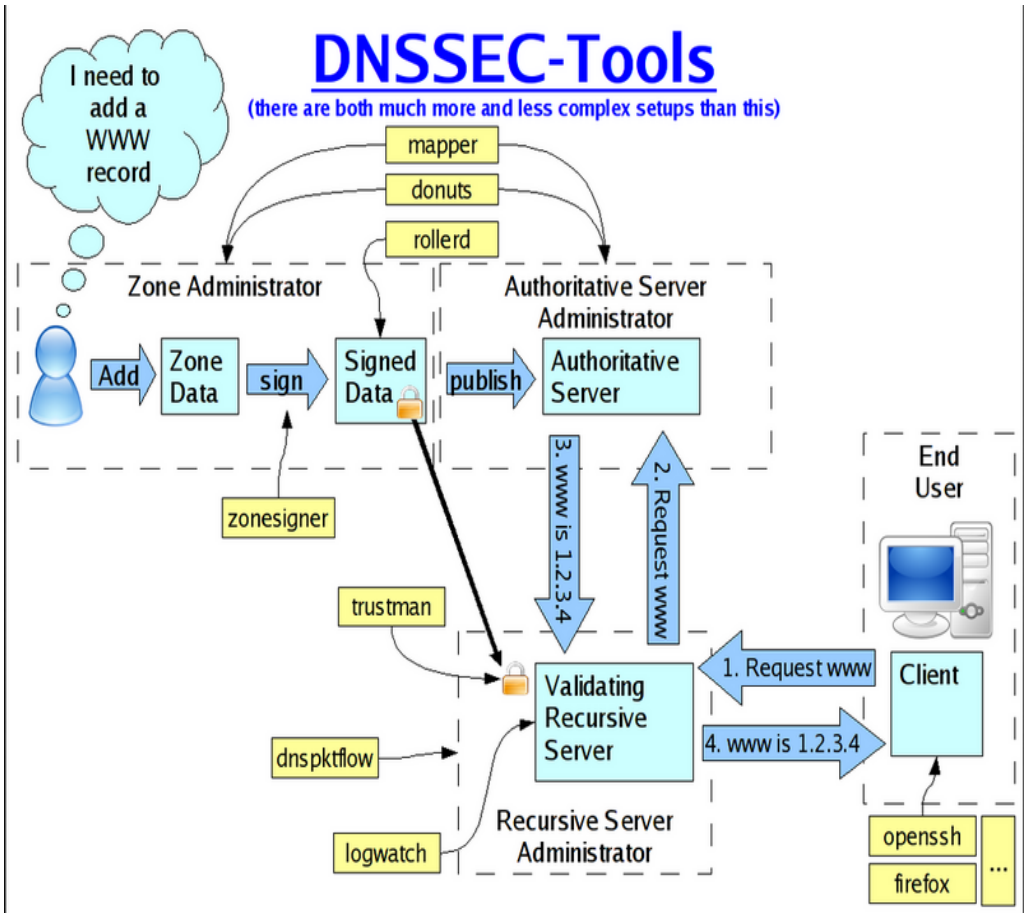




2.4 DNSSEC-Tools

2.4.1Introduction de DNSSEC-Tools

Pour faciliter la gestion de zones sous DNSSEC, des outils open-source existent : DNSSEC-tools. Il s'agit d'une suite libre d'outils. L'organisation modulaire de DNSSEC-tools permet de construire sa propre solution.



2.4.2 Les étapes pour DNSSEC-Tools

Pour commencer, il faut d'abord installer le DNSSEC-Tools sur ma machine virtuelle.

apt-get install DNSSEC-Tools

Pour signer ma propre zone, il faut le fichier de zone direct pour mon domaine.

$TTL 3600

congchen.fr. 1000 IN SOA ns.congchen.fr. admin.congchen.fr.(

5 ; serial
7200 ; refresh(2 hours)
3600 ; retry(1 hour)
604800 ; expire(1 week)
600 ; minimum(10 minutes)
)
1000 NS ns.congchen.fr.
1000 NS ns6.gandi.net.
ns 1000 IN A 5.23.44.83
www 1000 IN A 5.23.44.83

Afin de signer ce fichier de la zone, il faut utiliser **zonesigner**. Pour la première fois, nous avons besoin d'ajouter **-genkeys** option pour dire au **zonesigner** que nous voulons générer de nouvelles clefs pour la zone (KSK et ZSK)

```
root@cong-dns:/etc/bind/congchen.fr.dnssec-tools# zonesigner -genkeys -zone congch
en.fr ./congchen.fr

        if zonesigner appears hung, strike keys until the program completes
        (see the "Entropy" section in the man page for details)

Generating key pair.....................................................++++++ ...
.....................++++++
Generating key pair.......++++++ ......++++++
Generating key pair.........................+++ ..............................
.....................................................................+++
Verifying the zone using the following algorithms: RSASHA256.
Zone fully signed:
Algorithm: RSASHA256: KSKs: 1 active, 0 stand-by, 0 revoked
                      ZSKs: 1 active, 1 stand-by, 0 revoked

zone signed successfully

congchen.fr:
        KSK (cur) 35977  2048  05/26/16 (congchen.fr-signset-00003)
        ZSK (cur) 10803  1024  05/26/16 (congchen.fr-signset-00001)
        ZSK (pub) 18126  1024  05/26/16 (congchen.fr-signset-00002)

zone will expire in 30 days
DO NOT delete the keys until this time has passed.
```

Les fichiers .key et .private ont été crées automatiquement dans le répertoire **dnssec-tools**. Un fichier 'zonefile.signed' est généré pour signer la zone. Ce qui sont crées en même temps sont la Zone associée, Key Signing Keys (ZSKs/KSKs), les fichiers de keyset, le fichier de dsset et un fichier de la configuration pour zonesigner pour congchen.fr. Ces fichiers sont générés dans le même repertoire que le fichier de zone est signé.

Zonesigner offre un grand nombre d'options supplémentaires pour affecter la signature de fichier de zone. Le temps d'expiration pour la clé, le nom du fichier et locations tous peuvent être changés par la commande.

Kcongchen.fr.+008+20265.key

Kcongchen.fr.+008+20265.private

Kcongchen.fr.+008+35614.key

Kcongchen.fr.+008+35614.private

Kcongchen.fr.+008+49524.key

Kcongchen.fr.+008+49524.private

congchen.fr

congchen.fr.krf

congchen.fr.signed

dsset-congchen.fr.

Rollerd est un démon qui automatise le processus de renouvellement des clefs … Rollerd permet aux administrateurs de ne pas s'occuper de ce processus complexe.

Pour réussir à faire le **rollerd**, il faut d'abord creer un fichier .rrf à l'aide de **rollinit**. Le **rollinit** crée des nouvelles entrées pour un rollrec fichier. Ce rollrec fichier va être utilize afin de gérer le rollover des clés pour les zones nommés.

Il faut commencer par créer le fichier de configuration de Rollerd (fichier .rrf). Le fichier de configuration peut être généré avec l'outil rollinit, il s'agit en fait d'un fichier permettant de mémoriser l'état courant durant les renouvellement de clefs.

```
skip        "info rollrec"
            version         "2"

roll        "congchen.fr"
            zonename        "congchen.fr"
            zonefile        "congchen.fr.signed"
            keyrec          "congchen.fr.krf"
            kskphase        "0"
            zskphase        "0"
            ksk_rolldate    " "
            ksk_rollsecs    "0"
            zsk_rolldate    " "
            zsk_rollsecs    "0"
            maxttl          "0"
            display         "1"
            phasestart      "new"
            #optional records for RFC5011 rolling:
            istrustanchor   "no"
```

holddowntime "60D"

2.5 Système de configuration DNS pour la plateforme informatique

2.5.1 Architecture globale

Toute la configuration DNS pour la plate-forme est concentrée dans un répertoire : /etc/bind/zones. Ce répertoire contient un sous-répertoire par nom de domaine géré (par exemple plil.fr). Le répertoire contient aussi un Makefile général permettant de lancer la configuration pour tous les noms de domaine.

```
DIRECTORIES=$(shell find /etc/bind/zones -mindepth 1 -maxdepth 1 -type d | xargs -n1 basename)
#DIRECTORIES = congchen.fr ilot.org
DIRALL = $(DIRECTORIES:%=all-%)
DIRCLEAN = $(DIRECTORIES:%=clean-%)

all: $(DIRALL)
$(DIRALL):
        $(MAKE) -C $(@:all-%=%)

clean: $(DIRCLEAN)
$(DIRCLEAN):
        $(MAKE) -C $(@:clean-%=%) clean

.PHONY: $(DIRALL) $(DIRCLEAN)
.PHONY: all clean
```

Le Makefile global construit aussi le fichier de configuration du démon de renouvellement des clés (rollerd) et relance ce démon.

2.5.2 Répertoires pour les noms de domaine

Ces répertoires contiennent principalement des fichiers d'extension .hosts. Ces fichiers permettent de décrire des machines de façon compacte. Voici un exemple de fichier .hosts :

; Zone definitions
congchen.fr 5.23.44 2001 :07a8 :116e :0047
; Servers
ZABETH 83 00 :16 :3e :87 :38 :4e
ZABETH2 84 00 :16 :40 :87 :38 :4e

La première ligne donne le nom de domaine concerné, le préfixe IPv4 des adresses et le préfixe IPv6 des adresses. Ensuite chaque machine est décrite par son nom court, son suffixe IPv4 et son adresse Ethernet. A noter que l'adresse IPv6 est

calculée en concaténant le préfixe IPv6 et l'EUI-64 dérivé de l'adresse Ethernet.

A partir des fichiers .hosts sont calculés des fichiers de configuration au format bind pour la résolution nom vers adresse IPv4, nom vers adresse IPv6, adresse IPv4 vers nom et adresse IPv6 vers nom.

L'utilitaire qui réalise cette conversion s'appelle mkzone et son code est donné en annexe.

Voici les fichiers générés à partir du fichier .hosts ci-dessus.

Fichier congchen.fr.v4dir

ZABETH IN A 5.23.44.83
ZABETH2 IN A 5.23.44.84

Fichier congchen.fr.v6dir

ZABETH IN AAAA 2001 :07a8 :116e :0047 :0216 :3eff :fe87 :384e
ZABETH2 IN AAAA 2001 :07a8 :116e :0047 :0216 :40ff :fe87 :384e

Fichier congchen.fr.v4rev

83 IN PTR ZABETH.congcehn.fr.
84 IN PTR ZABETH2.congcehn.fr.

Fichier congchen.fr.v6rev

e.4.8.3.7.8.e.f.f.f.e.3.6.1.2.0 IN PTR ZABETH.congchen.fr.
e.4.8.3.7.8.e.f.f.f.0.4.6.1.2.0 IN PTR ZABETH.congchen.fr.

C'est à partir des fichiers au format bind que sont fabriqués les fichiers de zone, suffixé par .zone, déclarés dans le fichier principal de bind. Un fichier de zone peut regrouper plusieurs de ces fichiers (par exemple le fichier de résolution nom vers adresse IPv4 et le fichier de résolution nom vers adresse IPv6). Il est aussi parfois nécessaire d'ajouter un fichier contenant des enregistrements plus exotiques. Le regroupement se fait en utilisant la directive $INCLUDE.

Fichier congchen-dir.tzone

$INCLUDE LOCALDIR/congchen.fr.v4dir
$INCLUDE LOCALDIR/congchen.fr.v6dir
$INCLUDE LOCALDIR/congchen-dir.zone.extra

A noter que pour que zonesigner puisse gérer le numéro de version de la zone, l'enregistrement SOA est automatiquement ajouté en tête du fichier de zone (voir l'exemple d'enregistrement SOA ci-dessous).

$TTL 259200

```
@   IN   SOA  ns.congchen.fr. postmaster.congchen.fr. ( 0 21600 3600 2592000
2592000 )
      IN   NS    ns.congchen.fr
      IN   NS    ns.gandi.net
```

Le Makefile d'un répertoire de zone lance automatiquement l'utilitaire mkzone sur les fichiers .hosts, assure la création initiale des fichiers .zone et lance zonesigner pour signer les enregistrements. Vous trouvez un exemple de fichier .zone ci-dessous.

Fichier congchen-dir.zone

```
$TTL       259200
@ IN SOA ns.congchen.fr.  postmaster.congchen.fr. (96 21600 3600 259200     259200)
   IN  NS   ns.congchen.fr.
   IN  NS   ns6.gandi.net.
$INCLUDE  /etc/bind/zones/congchen.fr/congchen.fr.v4dir
$INCLUDE  /etc/bind/zones/congchen.fr/congchen.fr.v6dir
$INCLUDE  /etc/bind/zones/congchen.fr/congchen-dir.zone.extra
```

Après la première utilisation de la commande zonesigner, les clefs ZSK et KSK sont générées. Il y a une clef KSK et deux clefs de ZSK. L'une des deux clefs ZSK est utilisée et l'autre est réservée pour le prochain basculement de clefs.

Le Makefile vérifie s'il existe déjà des clés ou pas. Si elles existent, zonesigner est lancé sans l'option de génération de clés.

Dans le Makefile pour pouvoir appeler zonesigner il fallait trouver le nom de domaine correspondant au fichier .zone. Seul le fichier de configuration principal de bind permet de faire cette correspondence. Un utilitaire C pour analyser le fichier a été écrit (voir en annexe).

Le Makefile assure aussi la création initiale du fichier de configuration de rollerd, démon permettant de renouveler automatiquement les clefs.

Voici le Makefile pour les zones

```
# Some constants
MKZONES:=/root/DNS/mkzones
FINDZONE:=/root/DNS/findzone
LOCALPATH:=$(dir $(abspath $(lastword $(MAKEFILE_LIST))))

# Files to be processed
HOSTS_TEMPLATES = $(wildcard *.hosts)
HOSTS_FILES=$(HOSTS_TEMPLATES:.hosts=.v4dir)
$(HOSTS_TEMPLATES:.hosts=.v4rev)
```

```
HOSTS_FILES+=$(HOSTS_TEMPLATES:.hosts=.v6dir)
$(HOSTS_TEMPLATES:.hosts=.v6rev)
ZONE_TEMPLATES = $(wildcard *.tzone)
ZONE_FILES = $(ZONE_TEMPLATES:.tzone=.zone)
ZONE_SIGNED_FILES = $(ZONE_TEMPLATES:.tzone=.zone.signed)
ZONE_EXTRA_FILES = $(ZONE_TEMPLATES:.tzone=.zone.extra)
# Main target
all: $(HOSTS_FILES) $(ZONE_FILES) $(ZONE_SIGNED_FILES)

# Clean target
clean:
 rm -f *.v[46]dir *.v[46]rev

# Generic rules
%.v4dir %.v4rev %.v6dir %.v6rev: %.hosts
	$(MKZONES) $< > $@

%.zone: %.tzone %.tsoa
	( cat `basename $@ .zone`.tsoa ;                    \
	  sed -e "s#LOCALDIR/#$(LOCALPATH)#" < $< ) > $@ ;     \
	path=`pwd`/$@ ;                                  \
	zone=`$(FINDZONE) $$path` ;                         \
	if [ ! -z "$$zone" ] ; then                         \
	      rollinit -directory $(LOCALPATH)              \
	            -zonefile $@.signed $$zone              \
	       > $@.rrf ;                                   \
	fi

%.zone.signed: %.zone $(HOSTS_FILES) $(ZONE_EXTRA_FILES)
	# trouve le nom de zone grace a findzone `pwd`/$@      \
	path=`pwd`/$< ;                                  \
	zone=`$(FINDZONE) $$path` ;                         \
	if [ ! -z "$$zone" ] ; then
 # verifier qu'un fichier K$zone*.key existe     \
	      keyname=`echo K$$zone*.key | cut -f1 -d\ `;    \
	      if [ -f "$$keyname" ] ;                   \
	      then                                      \
	            zonesigner -zone $$zone $< ;          \
	      else                                      \
	            zonesigner -genkeys -zone $$zone $< ;   \
	      fi ; true ;                               \
	fi

.PHONY: clean
```

Conclusion

Ce stage m'a fait découvrir les domaines de la virtualisation de serveurs et le système de nommage d'Internet : DNS. J'ai aussi pu aborder la sécurisation d'Internet au travers du protocole DNSSEC.

Durant le stage j'ai du m'auto-former sur la théorie du système de nommage d'Internet et en particulier le protocole de sécurisation de DNS qui est assez complexe.

D'un autre coté, j'ai du m'améliorer sur la pratique d'Unix pour concevoir un système de configuration efficace du DNS. Il est maintenant possible de reprendre ce système de configuration pour l'implanter sur la machine de production.

Annexe

findzone.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define BIND9_CONFIG    "/etc/bind/named.conf.local"
#define LINE 1024

int main(int argc,char *argv[])
{
      FILE *stream;
      if(argc!=2){
            fprintf(stderr, "Syntax: %s <zone file>.\n",argv[0]);
            exit(EXIT_FAILURE);
      }
      char *filename=argv[1];
      stream=fopen(BIND9_CONFIG,"r");
      if(stream==NULL){
            fprintf(stderr, "Cannot open bind9 config file.\n");
            exit(EXIT_FAILURE);
      }
      char line[LINE];
      char zone[LINE];
      char argument[LINE];
                        strcpy(zone,"");
      while(fgets(line,LINE,stream))
      {
            sscanf(line," zone \"%[^\"]",zone);
            if(sscanf(line," file \"%[^\"]",argument)==1){
               if(strcmp(argument,filename)==0) printf("%s\n",zone);
              }
      }
      return 0;
}
```

mkzones

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_PATH       1024
#define MAX_LINE       1024
#define MAX_TOKEN      1024

void mkipv4dir(FILE *file,char *name,char *netv4,int ip1,int ip2,int ip3,int ip44
){
fprintf(file,"%-30s IN A %s",name,netv4);
if(ip1>=0) fprintf(file,".%d",ip1);
if(ip2>=0) fprintf(file,".%d",ip2);
if(ip3>=0) fprintf(file,".%d",ip3);
if(ip4>=0) fprintf(file,".%d",ip4);
fprintf(file,"\n");
}

void mkipv4rev(FILE *file,char *name,char *netname,int ip1,int ip2,int ip3,int ii
p4){
char reverse[MAX_TOKEN];
if(ip4>=0) sprintf(reverse,"%d.%d.%d.%d",ip1,ip2,ip3,ip4);
else if(ip3>=0) sprintf(reverse,"%d.%d.%d",ip1,ip2,ip3);
else if(ip2>=0) sprintf(reverse,"%d.%d",ip1,ip2);
else if(ip1>=0) sprintf(reverse,"%d",ip1);
fprintf(file,"%-30s IN PTR %s.%s.\n",reverse,name,netname);
}

char *enumeratev6(char *address,char direction){
static char result[MAX_TOKEN];
int i,j;
int offset;
if(direction<0) offset=strlen(address)-1; else offset=0;
unsigned char separator=0;
for(i=0,j=0;i<strlen(address);i++){
 if((direction<0 && separator==1)||(direction>0 && separator==4))
   { result[j++]=(direction==1)?':':'.'; separator=0; }
 if(address[offset+direction*i]!=':')
   { result[j++]=address[offset+direction*i]; separator++; }
 }
result[j]='\0';
```

```
return result;
}

void mkipv6dir(FILE *file,char *name,char *netv6,char *ip6){
char address[MAX_TOKEN];
fprintf(file,"%-30s IN AAAA ",name);
sprintf(address,"%s:%s",netv6,ip6);
fprintf(file,"%s\n",enumeratev6(address,1));
}

void mkipv6rev(FILE *file,char *name,char *netname,char *ip6){
fprintf(file,"%-30s IN PTR %s.%s.\n",enumeratev6(ip6,-1),name,netname);
}

char *getconfline(FILE *file){
static char line[MAX_LINE];
while(1){
 if(fgets(line,MAX_LINE,file)==NULL) return NULL;
 int i;
 for(i=0;i<strlen(line);i++) if(strchr(" \t\n\r",line[i])==NULL) break;
 if(i>=strlen(line)) continue;
 if(line[0]==';') continue;
 return line;
 }
return NULL;
}

int main(int argc,char *argv[]){
if(argc!=2){
 fprintf(stderr,"%s <hosts file>\n",argv[0]);
 exit(EXIT_FAILURE);
 }
char *path=argv[1];
FILE *file=fopen(path,"r"); if(file==NULL){ perror("open"); exit(EXIT_FAILURE);
}
char *period=strrchr(path,'.');
if(period!=NULL) *period='\0';
char path_v4dir[MAX_PATH]; sprintf(path_v4dir,"%s.v4dir",path);
char path_v4rev[MAX_PATH]; sprintf(path_v4rev,"%s.v4rev",path);
char path_v6dir[MAX_PATH]; sprintf(path_v6dir,"%s.v6dir",path);
char path_v6rev[MAX_PATH]; sprintf(path_v6rev,"%s.v6rev",path);
FILE *file_v4dir=fopen(path_v4dir,"w"); if(file_v4dir==NULL){ perror("open_v4dirr
"); exit(EXIT_FAILURE); }
FILE *file_v4rev=fopen(path_v4rev,"w"); if(file_v4rev==NULL){ perror("open_v4revv
```

```
"); exit(EXIT_FAILURE); }
FILE *file_v6dir=fopen(path_v6dir,"w"); if(file_v6dir==NULL){ perror("open_v6dirr
"); exit(EXIT_FAILURE); }
FILE *file_v6rev=fopen(path_v6rev,"w"); if(file_v6rev==NULL){ perror("open_v6revv
"); exit(EXIT_FAILURE); }
char *line;
if((line=getconfline(file))==NULL){ fprintf(stderr,"Void configuration file\n");;
 exit(EXIT_FAILURE); }
char netname[MAX_TOKEN],netv4[MAX_TOKEN],netv6[MAX_TOKEN];
netname[0]='\0'; netv4[0]='\0'; netv6[0]='\0';
sscanf(line,"%s %s %s",netname,netv4,netv6);
while((line=getconfline(file))!=NULL){
  char name[MAX_TOKEN];
  char addrv4[MAX_TOKEN];
 char addrv6[MAX_TOKEN];
  if(sscanf(line,"%s %s %s",name,addrv4,addrv6)!=3) continue;
  int ip1=-1,ip2=-1,ip3=-1,ip4=-1;
  char end[MAX_TOKEN];
  if(sscanf(addrv4,"%u.%u.%u.%u%s",&ip1,&ip2,&ip3,&ip4,end)==3 || sscanf(addrv4,,
"%u.%u.%u%s",&ip1,&ip2,&ip3,end)==3 ||
     sscanf(addrv4,"%u.%u%s",&ip1,&ip2,end)==2 || sscanf(addrv4,"%u%s",&ip1,end))
==1){
    mkipv4dir(file_v4dir,name,netv4,ip1,ip2,ip3,ip4);
    mkipv4rev(file_v4rev,name,netname,ip1,ip2,ip3,ip4);
    }
  char mac1[3],mac2[3],mac3[3],mac4[3],mac5[3],mac6[3];
  if(sscanf(addrv6,"%2[0-9a-fA-F]:%2[0-9a-fA-F]:%2[0-9a-fA-F]:%2[0-9a-fA-F]:%2[00
-9a-fA-F]:%2[0-9a-fA-F]%s",mac1,mac2,mac3,mac4,mac5,mac6,end)==6){
    char ip6[MAX_TOKEN];
    mac1[1] |= 0x02;
    sprintf(ip6,"%s%s:%sff:fe%s:%s%s",mac1,mac2,mac3,mac4,mac5,mac6);
    mkipv6dir(file_v6dir,name,netv6,ip6);
    mkipv6rev(file_v6rev,name,netname,ip6);
    continue;
    }
  char ip6[MAX_TOKEN];
  if(sscanf(addrv6,"%40[0-9a-fA-F:]%s",ip6,end)==1){
 mkipv6dir(file_v6dir,name,netv6,ip6);
    mkipv6rev(file_v6rev,name,netname,ip6);
    }
  }
fclose(file_v4dir); fclose(file_v4rev); fclose(file_v6dir); fclose(file_v6rev);
fclose(file);
return 0;
```

```
}
```