RAPPORT :

CREATION INFRASTRUCTURE LINUX

Membre du Groupe 2 :

Quentin GAVILAN

Brian LOK

KHALED MAHDI

MR VICTOR

10/12/2023

SOMMAIRE

INTRODUCTION 3
DNS 4
DHCP 5
FIREWALL 6
SITE WEB 7
PARTAGE DE DOCUMENT 8
SAUVEGARDE 9
POINT AMELIORATION 10
CONCLUSION 11
ANNEXE 12

INTRODUCTION

Nous avons l'honneur de vous présenter notre projet, conçu et développé par un groupe de trois étudiants en 1er année de Master. Notre objectif a été de créer une infrastructure d'entreprise robuste et fonctionnelle, basée sur Linux. Ce projet ambitieux vise à déployer un ensemble de services essentiels pour une entreprise moderne, tout en garantissant sécurité, efficacité et fiabilité.

Afin de répondre à cette tache notre équipe IT a intégré plusieurs services essentiels au fonctionnement de notre entreprise nommée KGB CORP favorisant ainsi un environnement de travail collaboratif et productif tout en gardant une sécurité suffisante, c'est pour cela que chaque service utilisera une distribution Debian connu pour sa stabilité et sa sécurité actif.

Nous avons donc implémenté :

- un service DHCP pour l'attribution automatique des adresses IP,
- un service DNS pour la résolution des noms de domaines
- un service WEB offrant une interface utilisateur intuitive et une porte d'entrée vers nos services.
- une protection avec un Pare-Feu garantissant une sécurité maximale contre les menaces extérieures.
- une connexion à Internet
- un service SSH, offrant un accès sécurisé et crypté au réseau de l'entreprise.
- un service de Sauvegarde et de Partage de Document

Ce projet représente non seulement une opportunité d'appliquer nos connaissances théoriques, mais aussi de développer des compétences pratiques dans la mise en place d'une infrastructure IT complète et moderne. Vous pourrez retrouver notre maquette de notre projet en annexe

Nous vous invitons à découvrir le travail réalisé tout au long de ce projet.

DNS

Notre entreprise **KGB CORP** nécessite l'installation d'un serveur **DNS** pour établir un **relais** entre les noms de domaine de notre **LAN** et ceux **d'Internet**. Nous déclarons donc un **nom de domaine** pour notre entreprise, **kgb.local**, et installons le paquet **Bind9**.

Pour assurer une **haute disponibilité** du serveur, nous mettons en place un système **Maître-Esclave**, impliquant la création de deux serveurs **DNS**. Le premier, faisant **autorité** sur une zone **DNS**, répliquera chaque changement effectué vers les serveurs **secondaires** (esclaves) de la **zone**.

Nous configurons ce serveur dans **/etc/bind/named.conf.local** pour y intégrer les détails de la zone **kgb.local** et sa zone inverse. Après cette configuration, nous créons deux fichiers de zone, **kgb.fw.zone** et **kgb.rev.zone**, contenant les enregistrements **SOA**, **A**, **NS**, et **PTR** nécessaires pour stocker les informations de la zone administrée, incluant les **numéros de version** et le **temps de rafraîchissement** des requêtes.

Afin de faciliter la **résolution** des **noms de domaine** du serveur vers Internet, il est nécessaire de modifier le fichier **/etc/bind/named.conf.options** pour y intégrer les adresses IP des serveurs **DNS** suivants, permettant ainsi d'accéder à la **racine** des noms de domaine.

Une fois le serveur principal configuré, nous procédons de même avec le **serveur secondaire**. Nous modifions le fichier **/etc/bind/named.conf.local**, en reprenant les **mêmes** inscriptions que dans le fichier maître, mais en changeant le type de serveur de **master** à **slave**, pour indiquer qu'il s'agit d'un serveur **secondaire**. Le serveur secondaire **vérifiera** régulièrement le **numéro de version** du fichier de configuration du serveur **maître** et, en cas de changement, les fichiers de configuration seront **transférés** sur le serveur secondaire, assurant ainsi une **cohérence** pour la zone **kgb.local.**

Nous **validons** cette configuration en **testant** la résolution des noms de domaine **kenny.kgb.local** et de l'adresse IP 192.168.0.40, obtenant des résultats positifs.

Retrouvez les fichiers de configuration complet de ce service dans la partie annexe

DHCP

Pour la configuration de notre **DHCP** installé sur notre machine **Debian**, nous avons opté pour l'installation du paquet **isc-dhcp-server**. Qui est utilisé pour mettre à disposition le service de **DHCP**, utilisé pour attribuer **automatiquement** des adresses **IP** et des informations réseaux tel que le **Gateway** ou le serveur **DNS**.

Ce dernier contient **3 composantes** clés que nous avons modifiées afin de déployer notre **DHCP** :

- **Configuration par défaut :** le **fichier « /etc/default/isc-dhcp-server »** permet au programme de connaitre l'emplacement de la configuration utilisé par notre **DHCP**, ainsi que l'interface réseau utilisé par le service ici qui est **enp0s3**
- **Fichiers de configuration** : Nous modifions par la suite le .conf de **« /etc/dhcp/dhcpd.conf** » afin de lui attribuer les informations réseaux pour **KGB CORP** tel que : **la plage ip = 192.168.0.1** à **192.168.0.254**, le **DNS = 192.168.0.50** et **192.168.0.40**, le **gateway = 192.168.0.254** et le **broadcast = 192.168.0.255**
- **Fichiers de journalisation** : Le serveur **DHCP** enregistre ses activités dans des fichiers **journaux globaux** tel que les **journaux système** pour le **dépannage** et la **surveillance**, nous pouvons aussi vérifier l'état du **DHCP** et les requêtes envoyés en exécutant la commande

systemctl status isc-dhcp-server

Une fois ces étapes effectuées nous pouvons **observer** après **redémarrage** du **service** le bon fonctionnement de notre **DHCP** en effectuant un test de requête DHCP avec un client, on remarque le bon déroulement du processus et de la mise en place de notre service avec les commandes **ip a** sur le client et **systemctl status isc-dhcp-server**

Retrouvez les fichiers de configuration et test complet de ce service dans la partie annexe

FIREWALL

Notre entreprise **KGB CORP**, traite des données **sensibles** et **hautement confidentielles**, nécessitant un **réseau protégé**, pour cela nous devons mettre en place un **pare-feu**, et créer des **règles spécifiques** afin de réguler les connexions sur le réseau. Nous avons choisi d'utiliser **« iptables »** et de créer des règles afin d'autoriser spécifiquement les connexions **HTTP, HTTPS, ICMP** (pour les tests de connexion intranet) et **DNS**, tant en **entrée** (input) qu'en **transit** (forward). Cette démarche vise à **sécuriser** le réseau tout en facilitant la communication sur ces **ports** essentiels.

Pour **renforcer** la sécurité, nous avons par **défaut bloqué** les **tables d'entrée** et de **transit**, acceptant uniquement les communications définies par des règles **spécifiques** du pare-feu. Une fois avoir protégé le **LAN** l'entreprise, **KGB CORP**, nécessite impérativement une **connexion internet** stable et sécurisée pour son environnement de **travail**, rôle également assuré par le **pare-feu.**

La machine **Debian**, configurée avec le **pare-feu**, est équipée de **deux cartes réseau** : l'une connectée à **Internet** et l'autre au **LAN**, faisant office de **routeur** pour basculer d'un réseau à l'autre. Cependant, pour que les utilisateurs accèdent à **Internet**, il est nécessaire que leur **IP privée** soit **convertie** en **IP publique**. Pour ce faire, nous utilisons le **NAT**, processus idéal pour cette tâche. Ainsi, une **règle NAT** est mise en place sur le pare-feu via **iptables** pour convertir les adresses IP de **192.168.0.0/24** en **10.0.2.4**, qui est l'IP de la seconde interface.

Une fois les règles appliquées nous testons la connexion vers internet et grâce à une configuration adéquate du service **DNS**, nous sommes en mesure de **pinger** des noms de domaines accessibles depuis **Internet** ainsi accéder a des pages WEB.

Retrouvez les règles de configuration du firewall dans la partie annexe

SITE WEB

L'entreprise KGB CORP, confrontée à la nécessité de se développer, envisage la création d'un site web pour promouvoir sa marque à l'échelle mondiale. Dans cet objectif, nous avons décidé d'établir un serveur Web sous Linux, en utilisant le logiciel Nginx, réputé pour son efficacité dans la gestion de nombreux clients simultanés, grâce à sa faible consommation de ressources.

Pour configurer ce logiciel, nous modifions les fichiers de configuration principaux situés dans le répertoire /etc/nginx/sites-available. Cependant, posséder un site Web ne suffit pas ; il est impératif de sécuriser le site. Chiffrer les données envoyées et reçues est un bon point de départ. À cet effet, nous passons d'une connexion HTTP sur le port 80 à HTTPS sur le port 443. Nous modifions donc le fichier de configuration pour indiquer au service d'écouter le port 443 et de charger la page en HTTPS de manière obligatoire. Nous imposons cette mesure pour la sécurité des utilisateurs, en écoutant sur le port 80 et en redirigeant la connexion vers le port 443 grâce à une erreur 301, indiquant la permanence du changement.

A présent pour finaliser la connexion HTTPS nous avons le choix entre créer des certificats auto-signés par le serveur, ce qui pourrait générer un message d'alerte de sécurité dans les navigateurs, ou obtenir un certificat authentique d'une autorité de certification.

Des tentatives non concluantes ont été effectuées pour établir une autorité de certification interne, qui n'a pas abouti en raison d'un manque de confiance des navigateurs, malgré la réussite de l'installation du serveur CA et de la génération du certificat SSL.

Malgré ces obstacles, le site web chiffre les données avant leur envoi.

Retrouvez les fichiers de configuration complet de ce service dans la partie annexe

PARTAGE DE DOCUMENT

Afin de permettre aux **utilisateurs d'échanger** des **fichiers** sans passer par un **cloud public** ou l'envoi par **email**, nous avons mis en place un **dossier commun** sur un **serveur distant**, accessible via **SFTP** (SSH File Transfer Protocol). Ce système nécessite que chaque utilisateur dispose d'un **identifiant** et d'un **mot de passe.**

La création d'un utilisateur d'accès étant donc **nécessaire** nous avons créé un utilisateur nommé **« depot »** avec un répertoire **chroot**. Cette approche **limite** les **accès** aux **utilisateurs** qui se connectent au **serveur**, leur permettant **uniquement** de **récupérer** ou de **déposer** des documents. Le répertoire **chroot** crée un environnement **isolé**, empêchant les utilisateurs **d'accéder** à d'autres parties du **système de fichiers** du **serveur** sécurisant son **environnement**.

Pour la configuration du **daemon SSH,** il est nécessaire de modifier le fichier de configuration pour autoriser uniquement les commandes **SFTP** et spécifier le répertoire **Chroot**. La particularité de ce répertoire chroot, étant donné que **« root »** est propriétaire de **/home/depot**, est que l'utilisateur **« depot »** ne peut pas **interagir** directement dans ce **répertoire** mais uniquement dans ses **sous-répertoires**. Par exemple, les répertoires **/home/depot/Client** et **/home/depot/Finance** peuvent être créés et attribués à l'utilisateur **« depot »,** lui permettant de gérer les fichiers dans ces dossiers.

Après la mise en place de cette **configuration**, les utilisateurs d'autres postes peuvent se **connecter** en **SFTP** en utilisant la commande :

sftp depot@adresse_ip_du_serveur

Une fois le **mot de passe** saisi, ils auront accès aux commandes **SFTP**, comme **cd** pour naviguer dans les répertoires ou **get** et **put** pour **télécharger** ou **déposer** des fichiers sur le **serveur**.

Retrouvez les fichiers de configuration complet de ce service dans la partie annexe

SAUVEGARDE

Pour la **sauvegarde**, un autre **serveur** est utilisé. Nous suivons donc les mêmes étapes que celles mentionnées précédemment pour créer un **utilisateur**, mais dans ce cas, cet utilisateur sera **exclusivement** utilisé par le **serveur** pour **déposer** la **sauvegarde** du dépôt.

Le **processus** est le suivant :

- Les répertoires **/home/depot** et ses **sous-répertoires** sont **compressés** en un seul fichier **zip**, qui est ensuite stocké dans **/etc/tmp.**
- Une connexion **SFTP** est établie entre le serveur de partage de fichiers et le serveur de **sauvegarde**.
- Le fichier **zippé** est **transféré** sur le serveur de sauvegarde, puis la **connexion** est **terminée**.

Les commandes **« expect »** sont utilisées pour automatiser la connexion et le transfert **SFTP** entre les **deux serveurs**. Cette automatisation est cruciale pour assurer la régularité et la fiabilité des **sauvegardes**.

La **procédure** automatisée est mise en œuvre en exécutant un **script** spécifique. Ce script est programmé pour s'exécuter à des intervalles réguliers grâce à l'utilisation du programme **« crontab ».** Le "crontab" est un programme système sous Unix permettant de planifier l'exécution de tâches (scripts ou programmes) à des moments précis. En utilisant la commande

crontab -e

Nous pouvons **modifier directement** le fichier **crontab** actuel pour ajouter ou modifier la **planification** de **tâches**.

Retrouvez le Script complet de ce service dans la partie annexe

POINT AMELIORATION

Malgré nos efforts déployés dans ce projet, nous avons identifié des améliorations potentielles de notre infrastructure, qui ne respecte pas entièrement les meilleures pratiques en matière de sécurité et de disponibilité des services.

Nous aurions pu envisager l'amélioration du service DHCP en instaurant une redondance grâce à un second serveur DHCP, augmentant ainsi la haute disponibilité du service, tout en renforçant la sécurité par le filtrage des adresses MAC, limitant l'accès aux seuls employés.

De plus, l'amélioration du service de sauvegarde et de partage de documents aurait pu être envisagée, notamment en renforçant la sécurité grâce à l'utilisation de clés RSA partagées entre les utilisateurs et les serveurs pour authentifier les connexions, et en ajoutant un second serveur de sauvegarde pour accroître la haute disponibilité du service.

Concernant le service de pare-feu, nous pourrions améliorer nos règles afin de restreindre davantage les connexions sur le LAN, renforçant ainsi la sécurité.

CONCLUSION

En conclusion, notre projet, illustre la création d'une infrastructure d'entreprise **robuste** et **fonctionnelle** sous **Linux,** spécifiquement conçue pour **KGB CORP**. Notre démarche a été guidée par l'objectif de **déployer** des services essentiels pour une entreprise moderne, tout en assurant un niveau de sécurité, **d'efficacité** et de **fiabilité importante**.

Nous avons mis en place **divers** services **clés**, notamment un service **DHCP** et **DNS** afin de rendre possible la création d'un **LAN**, ainsi qu'une **interface web sécurisée** et **optimisée** pour les **utilisateurs**. La **sécurité** est renforcée grâce à un **pare-feu** efficace donnant un accès à **Internet** et un accès réseau **sécurisé** et crypté via **SSH** pour chaque service. De plus, nous avons intégré des **services** innovants de **sauvegarde** et de **partage de documents**, en utilisant des méthodes **sécurisées** telles que **SFTP** et des systèmes de **sauvegarde automatisés** pour garantir la **protection** et la **disponibilité** des **données**.

Chaque aspect de ce projet a été **méticuleusement** conçu pour **favoriser** un **environnement** de **travail collaboratif** et **productif**, tout en maintenant une **sécurité rigoureuse**. L'utilisation de la distribution **Debian**, reconnue pour sa **stabilité** et sa **sécurité**, soutien notre engagement envers une **infrastructure fiable** et à jour.

Ce projet est le reflet de notre **capacité** à appliquer nos **connaissances théoriques** à des scénarios **pratiques**, tout en développant nos compétences dans la mise en place d'une infrastructure IT **complète** et **moderne**. Nous espérons que notre travail contribuera à faire de **KGB CORP** l'entreprise **qu'elle aspire à être**.

ANNEXE

Figure 1: Modification fichier config DNS maitre /etc/bind/named.conf.local

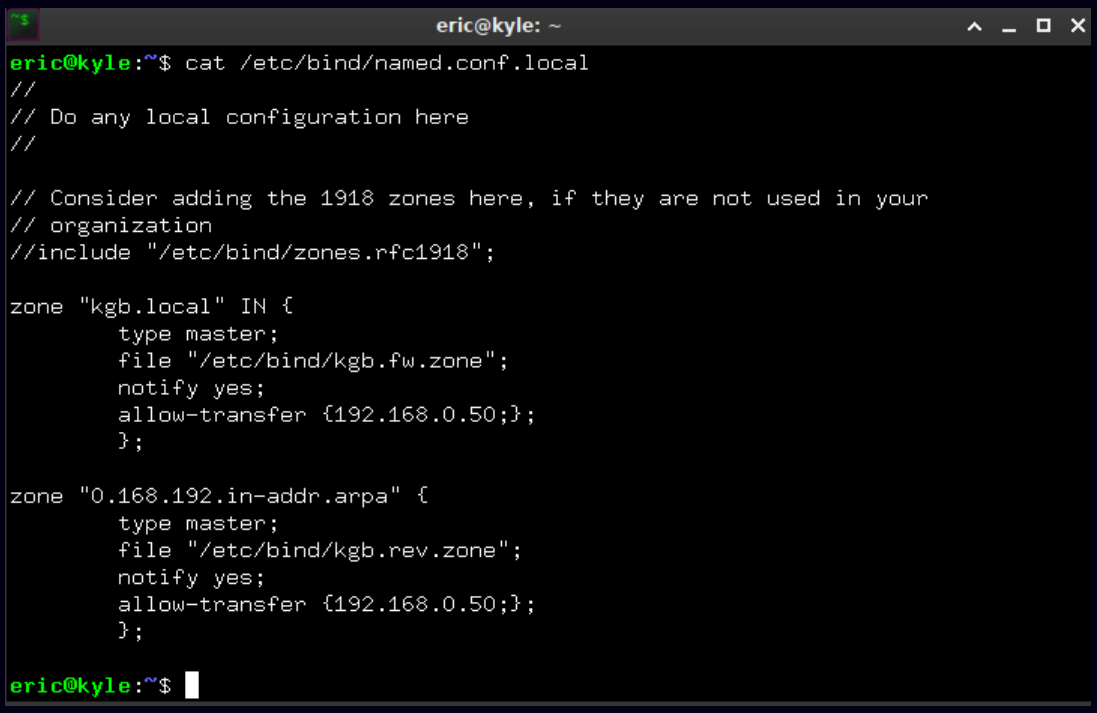

Figure 2 : Modification fichier configuration /etc/bind/kgb.rev.zone Maitre

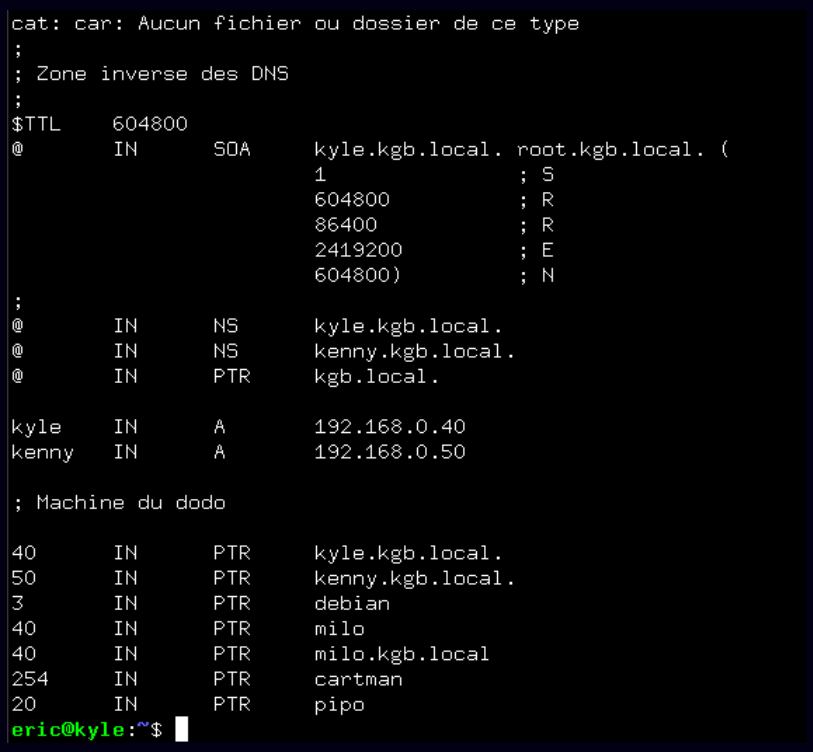

Figure 3: Modification fichier config /etc/bind/kgb.fw.zone Maitre

```
cat car /etc/bind/kgb.fw.zone
cat: car: Aucun fichier ou dossier de ce type
;
;
$TTL    604800
@       IN      SOA     kyle.kgb.local  root.kgb.local. (
                        20181226        ; Serial
                        604800          ; Refresh
                        86400           ; Retry
                        2419200         ; Expire
                        604800 )        ; Negative Cache TTL

;Servuer DNS

@       IN      NS      kyle.kgb.local.
@       IN      NS      kenny.kgb.local.
@       IN      A       192.168.0.40
@       IN      A       192.168.0.50

; Translation DNS

kyle    IN      A       192.168.0.40
kenny   IN      A       192.168.0.50

; Machine du Domaine

debian  IN      A       192.168.0.3
milo    IN      A       192.168.0.40
cartman IN      A       192.168.0.254
pipo    IN      A       192.168.0.20
eric@kyle:~$
```

Figure 4 : Test du client sur la zone inverse et résolution de nom

```
eric@kyle: ~
eric@kyle:~$ nslookup milo
Server:         192.168.0.40
Address:        192.168.0.40#53

Name:   milo.kgb.local
Address: 192.168.0.40

eric@kyle:~$ nslookup 192.168.0.40
40.0.168.192.in-addr.arpa       name = milo.0.168.192.in-addr.arpa.
40.0.168.192.in-addr.arpa       name = milo.kgb.local.0.168.192.in-addr.arpa.
40.0.168.192.in-addr.arpa       name = kyle.kgb.local.

eric@kyle:~$
```

Figure 5 : Fichier de configuration /etc/bind/named.conf.local de l'esclave

```
eric@kenny:~$ cat /etc/bind/named.conf.local
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";

zone "kgb.local"        IN      {
        type slave;
        file "/var/cache/bind/kgb.fw.zone";
        masters { 192.168.0.40; };
        };

zone "0.168.192.in-addr.arpa" {
        type slave;
        file "/var/cache/bind/kgb.rev.zone";
        masters { 192.168.0.40; };
        };
eric@kenny:~$
```

Figure 6: Test Client reprise de la partie esclave lors d'une mise hors tension du Maitre

```
eric@debian: ~
eric@debian:~$ sudo nano /etc/resolv.conf
[sudo] Mot de passe de eric :
eric@debian:~$ nslookup milo
Server:         192.168.0.50
Address:        192.168.0.50#53

Name:   milo.kgb.local
Address: 192.168.0.40

eric@debian:~$ sudo nano /etc/resolv.conf
eric@debian:~$ nslookup milo
;; communications error to 192.168.0.40#53: timed out
;; communications error to 192.168.0.40#53: timed out
;; communications error to 192.168.0.40#53: timed out
Server:         192.168.0.50
Address:        192.168.0.50#53

Name:   milo.kgb.local
Address: 192.168.0.40
;; communications error to 192.168.0.40#53: timed out
;; communications error to 192.168.0.40#53: timed out
;; communications error to 192.168.0.40#53: timed out

eric@debian:~$
```

Figure 7 : Modification fichier de configuration du service DHCP /etc/dhcp/dhcp.conf

```
eric@kyle: /etc/dhcp
eric@kyle:/etc/dhcp$ cat dhcpd.conf
ddns-update-style none;
# A slightly different configuration for an internal subnet.
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.1 192.168.0.254;
  option domain-name-servers 192.168.0.40;
  option domain-name-servers 192.168.0.50;
  option domain-name "kgb.local";
  option routers 192.168.0.254;
  option broadcast-address 192.168.0.255;
  default-lease-time 600;
  max-lease-time 7200;
}
eric@kyle:/etc/dhcp$
```

Figure 8: Fichier configuration /etc/default/isc-dhcp-server

```
eric@kyle: /etc/default
eric@kyle:~$ cd /etc/default/
eric@kyle:/etc/default$ ls
anacron        cron    hwclock          locale      nss      useradd
avahi-daemon   dbus    intel-microcode  named       openvpn
bluetooth      grub    isc-dhcp-server  networking  saned
console-setup  grub.d  keyboard         nginx       ssh
eric@kyle:/etc/default$ cat isc-dhcp-server
# Defaults for isc-dhcp-server (sourced by /etc/init.d/isc-dhcp-server)

# Path to dhcpd's config file (default: /etc/dhcp/dhcpd.conf).
DHCPDv4_CONF=/etc/dhcp/dhcpd.conf
#DHCPDv6_CONF=/etc/dhcp/dhcpd6.conf

# Path to dhcpd's PID file (default: /var/run/dhcpd.pid).
#DHCPDv4_PID=/var/run/dhcpd.pid
#DHCPDv6_PID=/var/run/dhcpd6.pid

# Additional options to start dhcpd with.
#       Don't use options -cf or -pf here; use DHCPD_CONF/ DHCPD_PID instead
#OPTIONS=""

# On what interfaces should the DHCP server (dhcpd) serve DHCP requests?
#       Separate multiple interfaces with spaces, e.g. "eth0 eth1".
INTERFACESv4="enp0s3"
INTERFACESv6=""
eric@kyle:/etc/default$ cat isc-dhcp-server
```

Figure 9: Test Client requête DHCP

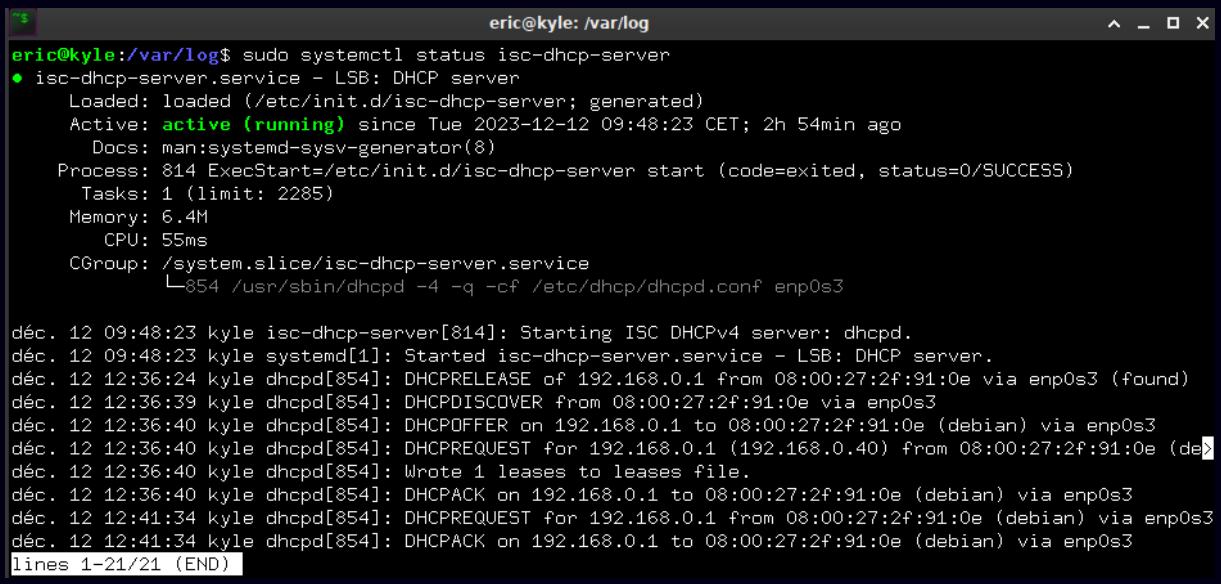

Figure 10 : Règle du firewall IPTABLES

```
[sudo] Mot de passe de eric :
# Generated by iptables-save v1.8.9 (nf_tables) on Tue Dec  5 10:37:09 2023
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [713:73900]
-A INPUT -p tcp -m tcp --sport 443 -j ACCEPT
-A INPUT -p tcp -m tcp --sport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -p tcp -m tcp --sport 53 -j ACCEPT
-A INPUT -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -p udp -m udp --sport 53 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A FORWARD -p udp -m udp --sport 53 -j ACCEPT
-A FORWARD -p tcp -m tcp --dport 53 -j ACCEPT
-A FORWARD -p udp -m udp --dport 53 -j ACCEPT
-A FORWARD -p tcp -m tcp --sport 80 -j ACCEPT
-A FORWARD -p tcp -m tcp --sport 443 -j ACCEPT
-A FORWARD -p tcp -m tcp --sport 53 -j ACCEPT
-A FORWARD -p tcp -m tcp --dport 443 -j ACCEPT
-A FORWARD -p tcp -m tcp --dport 80 -j ACCEPT
-A FORWARD -p icmp -j ACCEPT
COMMIT
# Completed on Tue Dec  5 10:37:09 2023
# Generated by iptables-save v1.8.9 (nf_tables) on Tue Dec  5 10:37:09 2023
*nat
:PREROUTING ACCEPT [466:38325]
:INPUT ACCEPT [6:748]
:OUTPUT ACCEPT [35:7664]
:POSTROUTING ACCEPT [179:17338]
-A PREROUTING -p tcp -m tcp --dport 1075 -j DNAT --to-destination 192.168.0.40
-A POSTROUTING -s 192.168.0.0/24 -o enp0s8 -j SNAT --to-source 10.0.2.4
COMMIT
```

Figure 11: Modification fichier config nginx /etc/nginx/site-availables/default

```
 GNU nano 7.2                          /etc/nginx/sites-available/default *
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        # SSL configuration
        #
        listen 443 ssl http2;
        listen [::]:443 ssl http2;

        ssl_certificate /etc/ssl/certs/MiloWeb.crt;
        ssl_certificate_key /etc/ssl/certs/milo.key;

        if ($scheme = http){
                return 301
                https://$server_name$request_uri;

        }
        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;

        server_name milo;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }

}
```

Figure 12 : Test de connexion du client au Site Web https://milo

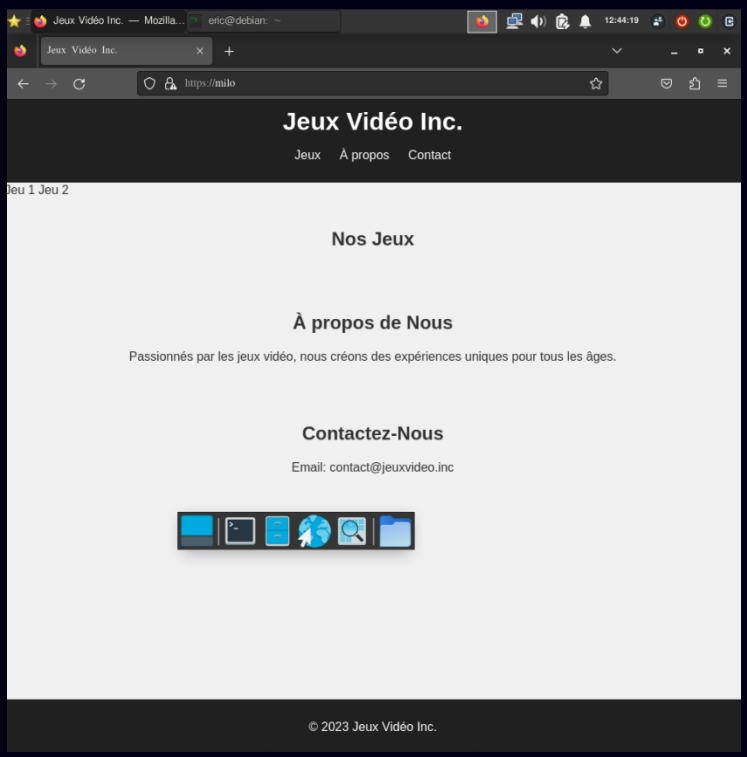

Figure 13: Configuration du fichier /etc/ssh/sshd_config (ligne 123 à 125)

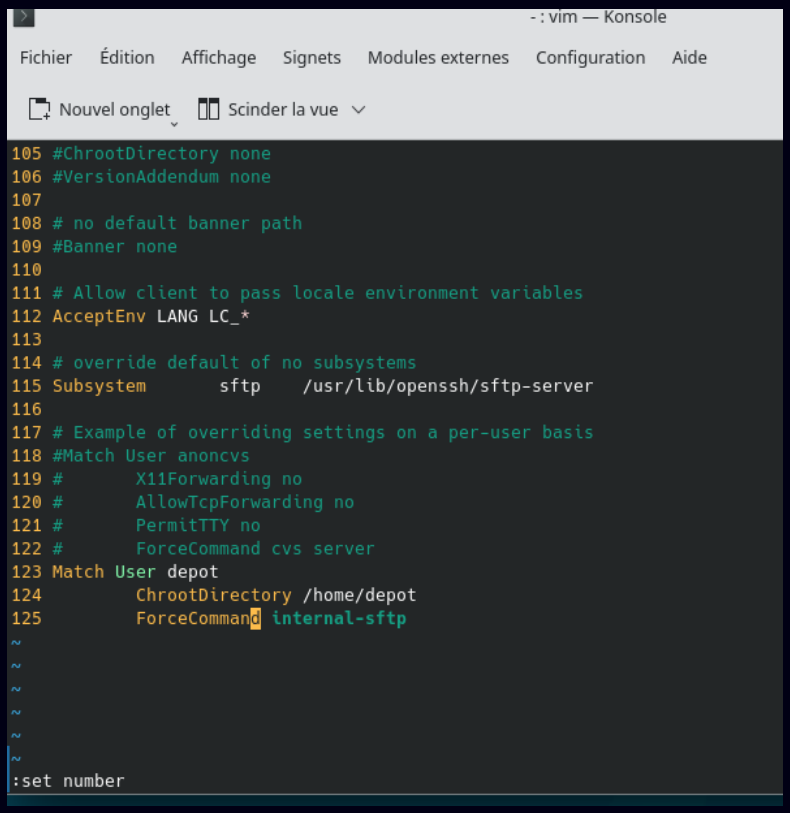

Figure 14: Script pour la création des répertoires dont l'utilisateur "dépot" sera propriétaire pour le SFTP

```
1  sudo mkdir -m 755 /home/depot/RH /home/depot/Finance /home/depot/Client
2
3  sudo chown depot:depot /home/depot/RH /home/depot/Finance /home/depot/Client
4
```

Figure 15: Script sftp_backup dans le répertoire /home/eric

```
nom_fichier_zip="$(date +'%d_%m_%y_%Hh%M_backup')"

tar -czf /tmp/$nom_fichier_zip /home/depot 2>/dev/null

expect <<EOF
spawn sftp depot@192.168.0.40
expect "password:"
send "123\r"
expect "sftp>"
send "put /tmp/$nom_fichier_zip /Sauvegarde\r"
expect "sftp>"
send "bye\r"
EOF

```

Figure 16 : Test de connexion du client au Site Web https://milo

```
backup_tache="*/5 * * * * /home/sftp_backup.sh"

(crontab -l 2>/dev/null; echo "$backup_tache") | crontab -
```