Présentation et Installation

Document 1

VERGEOT Enoal

IUT Réseaux & Télécoms

SAE203

Table des matières

Présentation générale de la solution technique 4

Installation de la solution 6

2.1 Prérequis techniques 6

2.2 Déploiement des fichiers 6

2.3 Installation des dépendances Python 6

2.4 Création et configuration du fichier de configuration 6

Présentation générale de la solution technique

Dans le contexte d'une entreprise répartie sur plusieurs sites, chaque site exploite son propre réseau local pour relier postes, serveurs et équipements spécifiques. Pour que chaque appareil obtienne une adresse IP de manière automatique, le protocole DHCP (*Dynamic Host Configuration Protocol*) est utilisé, un choix répandu pour éviter des configurations manuelles fastidieuses.
Cependant, pour garantir un accès maîtrisé et prévenir tout branchement non autorisé, seules certaines machines peuvent recevoir une adresse fixe. Cela suppose que chaque poste soit préalablement inscrit par l'équipe réseau, en utilisant son identifiant matériel (MAC address).
Dans la pratique, cela implique pour l'administrateur plusieurs points à surveiller :

- Pouvoir gérer l'ensemble des baux DHCP à partir d'un point unique, sans devoir se connecter individuellement à chaque serveur.
- Disposer d'un moyen simple pour ajouter ou retirer un poste autorisé.
- Vérifier régulièrement que la configuration reste propre, sans doublons d'adresses ou erreurs de saisie.

Pour répondre à ces contraintes, un ensemble d'outils développés en Python a été mis en place. L'idée principale est de centraliser la supervision et de simplifier les tâches courantes tout en maintenant un niveau de contrôle satisfaisant.
Les fonctionnalités couvertes incluent notamment :

- La surveillance à distance de l'état des serveurs DHCP à partir d'un serveur central unique.
- La gestion des baux (création, suppression, consultation, vérification) grâce à des commandes claires et explicites.
- Un contrôle automatique permettant de signaler toute adresse IP ou MAC incohérente, et de prévenir d'éventuels doublons.
- Une application rapide des changements grâce à un redémarrage maîtrisé du service dnsmasq.

Cette solution permet à l'équipe réseau de travailler avec :

- Moins de pertes de temps sur des actions répétitives.
- Une configuration alignée sur l'ensemble des sites, pour éviter les écarts.
- Un risque d'erreur réduit, grâce à des scripts conçus pour gérer les vérifications de base à leur place.

Les aspects techniques essentiels sont donc :

- **Une connexion sécurisée par SSH**
  Les scripts Python s'appuient sur Fabric, une bibliothèque bien connue pour exécuter des commandes à distance. Les échanges sont protégés par une clé RSA couplée à une phrase secrète, ce qui ferme la porte aux accès non autorisés.
- **Des actions strictement encadrées**
  Les commandes autorisées sur les serveurs DHCP sont limitées et définies précisément via les fichiers *authorized_keys* et *sudoers*. De cette manière, même en cas d'intrusion, l'impact reste fortement restreint.
- **Une gestion modulaire via dnsmasq**
  Chaque correspondance entre une adresse MAC et une IP est conservée dans un fichier distinct, rangé dans */etc/dnsmasq.d*. Cette organisation rend les modifications plus sûres : un ajout ou un retrait ne risque pas de compromettre la configuration générale.
- **Un routage centralisé et NAT**
  Le serveur principal remplit aussi la fonction de passerelle NAT pour les sous-réseaux VLAN simulant les différents sites. Les clients accèdent ainsi à Internet tout en restant logiquement isolés selon leur site.

Au final, l'entreprise bénéficie donc de plusieurs avantages concrets :

- Des baux DHCP fiables, ce qui écarte les coupures dues à des conflits d'adresses.
- Une gestion centralisée qui réduit considérablement les interventions dispersées.
- Une solution légère, déployable sur tout système Debian/Linux, sans nécessiter d'environnement graphique.
- Un contrôle d'accès rigoureux, qui renforce la sécurité globale du réseau.

Installation de la solution

2.1 Prérequis techniques

Avant d'installer la solution, certains prérequis doivent être respectés :

- Système d'exploitation recommandé : Debian ou une distribution Linux équivalente.
- Serveur central et serveurs DHCP configurés avec une mémoire minimale de 512 Mo, sans interface graphique pour limiter la consommation de ressources.
- Réseaux distincts simulés à l'aide de VLANs ou d'interfaces virtuelles.
- Connexion SSH configurée avec une clé RSA protégée par une phrase de passe.
- Fichier sudoers configuré pour autoriser l'exécution des commandes nécessaires sans mot de passe tout en limitant strictement les droits de l'utilisateur dédié.

2.2 Déploiement des fichiers

Sur le serveur central :

1. Créer un répertoire dédié pour les fichiers sources, par exemple : /home/*user*/src
2. Copier les fichiers suivants dans ce répertoire :
    a. Scripts Python :
        i. add-dhcp-client.py
        ii. remove-dhcp-client.py
        iii. check-dhcp.py
        iv. list-dhcp.py
    b. Modules Python communs :
        i. Dhcp.py
        ii. Config.py
        iii. validation.py

2.3 Installation des dépendances Python

1. Mettre à jour les paquets et installer pip :

```
sudo apt update
 sudo apt install python3-pip
```

2. Installer les bibliothèques Python nécessaires :

```
pip3 install fabric pyyaml
```

Remarque : Il est déconseillé d'utiliser Anaconda sur le serveur central en raison de la consommation mémoire trop élevée pour une machine légère.

2.4 Création et configuration du fichier de configuration

Créer un fichier de configuration YAML nommé par exemple config_superviseur.yaml et y insérer le contenu suivant en adaptant les IP à votre architecture :

```
---
dhcp_hosts_cfg: /etc/dnsmasq.d/hosts.conf
 user: dhcp-mod
```

dhcp-servers:
10.20.1.5: 10.20.1.0/24
10.20.2.5: 10.20.2.0/24