NETWORKPASS : HOW TO

Ce document à pour but la création de nouveaux modules en fonction des futurs matériels réseaux achetés, il existe déjà deux scripts pour des matériels CISCO et LINUX.

Si la société était amenée à acheter de nouveaux matériels d'une marque différente ou nécessitant un scénario différent de ceux existant, il faudrait créer un nouveau script secondaire dédié à ce nouveau matériel.

Ce document en explique la méthode.

Création d'un nouveau script pass_*****_change.py

Les étoiles dans ce document seront remplacées par le type de matériels réseaux, linux, cisco, zyxel…. Celui-ci sera défini dans le listing .csv et permettra l'ouverture du bon script.

Le fichier CSV s'écrit de la façon suivante :

server,ip,user,pw_user,pw_user,old_pw

- Argument 1 : server : correspond au type de matériel réseau
- Argument 2 : ip : adresse IP de du matériel
- Argument 3 : user : login de connexion
- Argument 4 : pw_user : mot de passe de connexion
- Argument 5 : pw_suser : mot de passe root si besoin pour la modification
- Argument 6 : old_pw

Exemple d'un ajout de ligne dans le csv

- zyxel,192.168.100.100,admin,admin,root,oldpw

Avec l'argument 1 : zyxel, le script principal appellera le script :

- pass_zyxel _change.py

ETAPE 1 :

Créer un fichier « pass_*****_change.py » dans le dossier NetworkPass

En rouge commentaire pour le document, à ne pas intégrer au code

ETAPE 2 :

Structure du code :

```
#! /usr/bin/env python
# -*- coding : Utf-8 -*-

"""Script de modification du mot de passe via SSH pour matériel *****"""
```

Import des modules nécessaires au bon fonctionnement du script

```
import sys #options système
import socket #gestion de la lecture des ports
import time #mise en place de temporisation
import csv #gestion des fichiers csv
import paramiko #gestion de la connexion SSH
#Récupération des arguments du fichiers CSV envoyés par le script principal
IP_ADDRESS = sys.argv[1]    adresse IP du matériel
USERNAME = sys.argv[2]      nom d'utilisateur de connexion
PASSWORD = sys.argv[3]      mot de passe de l'utilisateur de connexion
NEW_PW = sys.argv[4]        nouveau mot de passe généré
JOURNAL = sys.argv[5]       journal de bord de la mise à jour
PW_ROOT = sys.argv[6]       mot de passe super utilisateur, si besoin, sinon ne par
                            écrire cette ligne
```

Cette partie est commune à tous les matériels réseaux

```
#Connexion SSH et modification du mot de passe
SSH_CLIENT = paramiko.SSHClient()
SSH_CLIENT.set_missing_host_key_policy(paramiko.AutoAddPolicy())
Try: #Ouverture de la connexion
   SSH_CLIENT.connect(IP_ADDRESS, 22, USERNAME, PASSWORD, look_for_keys=False,
allow_agent=False)
#Gestion de l'erreur d'authentification
except paramiko.AuthenticationException:
   RESULT = "Authentication failed"
   with open(JOURNAL, "a") as suivi: #écriture dans le journal
      CSV_WRITER = csv.writer(suivi)
      CSV_WRITER.writerow([IP_ADDRESS, RESULT])
```

La ligne suivante arrête immédiatement le script est renvoi l'erreur 1 au script principal, celui-ci va interpréter l'erreur 1 est inscrire un problème au niveau de l'authentification lors de la connexion au matériel

```
sys.exit(1) #arrêt du script et renvoi de l'erreur
#Gestion de l'erreur de port (matériel non joignable)
except socket.error:
   RESULT = "Socket error"
with open(JOURNAL, "a") as suivi: #écriture dans le journal
   CSV_WRITER = csv.writer(suivi)
   CSV_WRITER.writerow([IP_ADDRESS, RESULT])
```

La ligne suivante arrête immédiatement le script est renvoi l'erreur 1 au script principal, celui-ci va interpréter l'erreur 2 est inscrire un problème au niveau de la connectivité avec le matériel (mauvaise IP, câble débranché)

```
sys.exit(2) #arrêt du script et renvoi de l'erreur
#Connexion réussie
else:
   REMOTE_CONNECTION = SSH_CLIENT.invoke_shell()
   RESULT = "Authentication succeeded"
```

A partir de maintenant viennent les lignes de commandes permettant le changement de mot de passe, il faut effectuer plusieurs scénarios de connexion SSH pour connaitre les lignes de commandes à exécuter. Pour la structure, c'est toujours la même (s'aider des scripts pass_cisco_change.py et pass_linux_change.py pour l'écriture du scénario)

```
   REMOTE_CONNECTION.send("ligne de commande\n") \n vaut Entrée
   time.sleep(1) à modifier en fonction du scénario
```

A répéter autant de fois que nécessaire pour modifier le mot de passe
Il est possible d'ajouter des conditions en fonction des scénarios
Une fois toutes les lignes du scénario écrites, on repasse sur un tronçon commun à tous avec la création de la variable FINAL

```
   FINAL = "Le mot de passe du matériel {1} a bien été changé : {0}".format(NE
W_PW, IP_ADDRESS)
```

l'écriture du résultat dans le journal

```
with open(JOURNAL, "a") as suivi: #écriture dans le journal
CSV_WRITER = csv.writer(suivi)
CSV_WRITER.writerow([IP_ADDRESS, RESULT, FINAL])
```

et la fermeture du script, avec la ligne suivante le script renvoi au script principal l'information que tout c'est bien passé

```
sys.exit(0)
```