Documentation Technique

Liste package necessaire :

Service utilisés :

1. MariaDB 10.5.12

    1. Fichier Configuration

Fichier configuration /etc/mysql/mariadb.conf.d/50-server.cnf Modifier la ligne

```
bind-address            = 127.0.0.1
```

en

```
bind-address            = 0.0.0.0
```

2. Configuration du pare-feu

```
ufw allow from 0.0.0.0 to any port 3306 >> /tmp/install.log #autoriser
depuis n'importe quelle ip une connexion sur le port 3306

iptables -A INPUT -p tcp --dport 3306 -j ACCEPT # Autorise les paquets en
entré sur le port 3306

iptables -I INPUT -p tcp --dport 3306 -i eth0 -m state --state NEW -m
recent --set >> /tmp/install.log

iptables -I INPUT -p tcp --dport 3306 -i eth0 -m state --state NEW -m
recent  --update --seconds 300 --hitcount 4 -j DROP  >> /tmp/install.log
#Les deux règles précédentes permettent de se protéger face à un
bruteforce, en effet si il y a plus de 5 paquet en entrée toute les 5
minutes, les paquets suivant seront jetés ainsi un bruteforce sur le server
mysql n'est pas possible
```

3. Script python Stegano

```
"""
This programs aims to convert insert in a png or jpeg file a message using
```

```
lsg of each byte

usage python3 encrypt input_file output_file message


"""

from os import putenv
from PIL import Image
import numpy as np
import sys
import hashlib


def convert_msg_to_binary(msg):
   """
   Convertis avec une expression compréhension n'importe quelle string en binaire en enlever le 08b pouvant appaitre lors de la conversion pour qu'il ne reste des les bit
   """
    return ''.join(["{:08b}".format(ord(x)) for x in msg])


def hashage(msg):
    """
    Fonction de hasahge, prend en entrée une string et renvois une autre string hashé en SHA1
    """
    hash = hashlib.sha1(msg.encode())
    hex_hash = hash.hexdigest()
    print(hex_hash)
    return hex_hash


def stegano(input_file, output_file, msg):

    img = Image.open(input_file)
    #on ouvre avec la librairie image l'image passée en entrée

    bin_msg = convert_msg_to_binary(msg)
    #On coonvertis le message que l'on veut caché en binaire

    bin_msg_lenght = len(convert_msg_to_binary(msg))
    #On récupère la longueur du message en binaire

    width, height = img.size
    #On récupère les dimensions de l'image

    img_data = np.array(img)
    #On recupère sous forme de tableau toutes les données de l'image grâce à numpy

    img_data = np.reshape(img_data, width*height*4)
    # Au lieu d'avoir un tableau de tableau de 4 octet on transforme le
```

```
tout en un seul grand tableau


    """
    Pour chaque bit du message en binaire, on regarde si le bit de point
faible de l'image correspondant  à la même valeur, si non on change le bit
de point faible de l'octet afin qu'il corresponde (via un modulo 2 car seul
le dernier bit d'un octet permet de savoir si la valeur décimale sera paire
ou impaire).

    A la fin les bit de points faibles des n (étant la longueur du message
en binaire ) premiers octets de l'image correspondront a ceux du message en
binaire
    """
    for i in range(len(bin_msg)):

        if(bin_msg[i] and not img_data[i] % 2):
            img_data[i] = img_data[i] - 1
        if((bin_msg[i] == '0') and (img_data[i] % 2)):
            img_data[i] = img_data[i] - 1


    img_data = np.reshape(img_data, (height, width, 4))
    #On recréer un tableau correspondant au format de l'image (tableau de
tableau de 4 octet)

    new_img = Image.fromarray(img_data)
    # On recréer une image via la librairie Image

    new_img.save(output_file)
    # On sauvegarde la nouvelle image avec le nom passé en entrer


def main(argv):

    message = ""
    msh_lenght = len(argv) - 3

    """
    Cela est utilie uniquement si l'utilisateur à passer en entrée un
message avec plusieurs mots afin de prendre en compte les espaces
    """
    for i in range(3, len(argv)):
        message = message + argv[i]


    # On appel la fonction stégano avec comme premier argument Le fichier
d'entré le second le fichier de sortie et le message hashé en dernier
    stegano(argv[1], argv[2], hashage(message))



#Main classique en python
if __name__ == "__main__":
```

```
    main(sys.argv)
```

- Instalation MariaDb en script (Utilisation de Except)

```
MYSQL_INSTAL=$(expect -c "

set timeout 5
spawn mysql_secure_installation
expect \"Enter current password for root:\"
send \"$MYMSG\r\"
expect \"Would you like to setup VALIDATE PASSWORD plugin?\"
send \"n\r\"
expect \"Change the password for root ?\"
send \"n\r\"
expect \"Remove anonymous users?\"
send \"y\r\"
expect \"Disallow root login remotely?\"
send \"n\r\"
expect \"Remove test database and access to it?\"
send \"y\r\"
expect \"Reload privilege tables now?\"
send \"y\r\"
expect eof
")
```

- Ajout d'utilisateur via sccript expect :

```
ADD_USER=$(expect -c "
set timeout 5
spawn adduser user
expect \"New password:\"
send \"azerty\r\"
expect \"Retype new password:\"
send \"azerty\r\"
expect \"Full name []:\"
send \"\r\"
expect \"Room Number []:\"
send \"\r\"
expect \"Work Phone []:\"
send \"\r\"
expect \"Home Phone []:\"
send \"\r\"
expect \"Other []\"
send \"\r\"
expect \"Is the information correct? []\"
send \"Y\r\"
```

```
expect eof
")
echo "$ADD_USER"
```

- Utilisattion de cron

Cron est un démon tournant en permanence, il va regarder dans ses règles crontab et exécuté tout les x temps (selon ce qu'on lui indique) des commandes. Voilà l'exemple d'une règle crontab

```
* * * * * /bin/bash /home/debian/exec_all_files.sh
```

Ici toute les minutes le script `exec_all_files` sera exécuté

Voici le tableau de à quoi correspond chaque `*`

*	*	*	*	*
minutes	heures	jour dans le mois	mois	jours dans la semaine

Par exemple pour faire toutes les 5 miutes il faut faire non pas

5 * * * * qui correspond a chaque fois que la minute 5 apparait (chaque heure) mais il faut faire */5 * * * * qui fera toute les 5 minutes le script.

Pour plus de détails afin de savoir comment gérer cron vous pouvez consulter ce site qui permet de savoir quels règles appliquer https://crontab.guru/every-1-hour

Nom script	Paths	text ?
show_passwd	/usr/shr/(un truc dans les script partagé avec le bon chmod	A voir si contenue où juste utilisation lien symbolique dans cron