Projet Linux

Presentation de l'idée

Tout d'abord, nous avons choisis de faire quelque chose d'un peu plus original que juste mettre en place un serveur. Mais nos connaissances en Linux n'étant pas immense nous ne pouvions pas nous lancer dans la réalisation de quelque chose de trop compliqué. C'est pourquoi nous avons choisis de faire un script shell qui demanderait de la recherche et de la connaissance en Linux.
Pour l'idée du script, nous avons cherché sur internet, puis après quelque clics nous avons eu l'idée de créer une protection de reseau wifi.
On fournit au script une liste des adresses MAC supposées se trouver sur notre reseau, le script se lance et nous informe si d'autres adresses MAC intrues sont connectées ou non et nous dit lesquelles.

C'est quoi une adresse MAC ?

Comme défini dans le glossaire Panoptinet, l'adresse MAC est l'**identifiant matériel** associé à une carte réseau. C'est un numéro unique définit par les constructeurs et composé de douze caractères hexadécimaux. Exemple : 00:07:CB:C4:48:C9. Deux ordinateurs équipés de carte réseau ne peuvent donc pas avoir la même adresse MAC, ce qui permet d'organiser un filtrage, machine par machine.

Le cheminement de notre recherche

La premiere étape à été de déterminer quelles addresses MAC sont connectés au reseau WIFI.
Pour cela, nous devons passer par deux étapes : récuperer l'IP du pc afin ensuite de recuperer toutes les addresses MAC connectés à notre wifi.
Pour connaitre l'ip nous avons rapidement trouvé en cherchant que la commande à utiliser était ***ifconfig*** ensuite, sur la deuxieme ligne, sur la deuxieme colonne se trouve notre addresse IP pour retrouver cette addresse on doit utiliser la commande ***awk***.

Awk est une commande complexe, ici ous l'avons utiliser pour extraire une suite de chiffres composant notre addresse IP. En effet awk est un langage a part entiere que

nous avous voulue etudié grace a ce script shell. Nous avons donc apris a extraire une ligne ou encore une colonne d'un paragraphe grace a cette commende très puissante.

Puis ensuite nous avons eu besoin de changer les derniers chiffres de cette adresse IP par « * ». Cela est dû à la définition d'une adresse IP. Les premiers chiffres désignent le réseau et les derniers désignent l'ordinateur, or nous voulons seulement le réseau donc les derniers chiffres n'importent pas.

Maintenant on passe à la deuxième partie : récupérer les adresses MAC présentent sur le réseau. Pour cela on déclare un tableau a l'aide de la commande ***declare –a*** et on nomme ce tableau ***MAC***. Ensuite on utilise une commande qui doit être téléchargée au préalable ***nmap –sP***. Cette commande récupère toutes les adresses en ligne sur le réseau et nous donne leur adresse MAC. Le scripte doit être lancé en root car nmap demande les droits pour que la totalité des informations soit affiché

On complète cette commande par un ***grep MAC*** qui récupère les lignes ou le mot MAC est écrit puis grâce à un ***awk '{ print $3}'*** donne la 3ème colonne et à l'aide de ***sort***, on classe les adresses MAC dans le tableau.

Maintenant on doit trouver toutes les adresses MAC dans un fichier, or on laisse à l'utilisateur la possibilité de laisser des commentaires et des espaces donc on doit les supprimer avant de classer les adresses MAC. Pour cela on utilise encore une fois la commande ***sed -e '/^[ ]*$/d' -e '/#/d '$1*** qui permet de supprimer tous les espaces et tous les commentaires dans le fichier envoyé dans la ligne de commande. Pour trouver cette commande notre démarche à été la suivante :

Pour effacer les lignes comprenant #
sed '/#/d' toto.txt

Pour effacer seulement le lignes commençant par un # (le $ marque le début de la ligne)
sed '/$#/d' toto.txt

Pour effacer les lignes vide sachant que le marqueur de fin de ligne est ^
sed '/$^/d' toto.txt

Pour effacer vide ou comprenant uniquement des espaces,
1) on défini un ensemble [ ] comprenant un seul caractère l'espace
2) on veut répéter les caractères N fois (N pouvant être nul) on utilise * (raison pour utiliser ' et non le " en bash)

sed '/$[ ]*^/d'

Pour faire plusieurs opérations on utilise -e, ce qui donne pour effacer les ligne commençant par # et les lignes vides ou comprenant uniquement des espaces

sed -e '/^[ ]*$/d' -e '/#/d' toto.txt

L'utilisateur peut choisir d'ajouter la commande ***–v*** si cette commande est entrée, les adresses mac connectés et celles écrie dans le fichier texte vont s'afficher grâce à une boucle for, dans le cas contraire seul les adresses MAC non présent dans le fichier texte vont être affiché.

On doit maintenant comparer les deux tableaux pour vérifier si des intrus sont présents. On met en place une double boucle for qui vérifie si les adresses connectées au réseau sont présentes dans le tableau ***MAC*** sont présentes dans les adresses autorisées de ***MACKNOW***. Si –v est activé alors chaque adresses présente et autorisée va être affichée, sinon seul les adresses présentes et non autorisées vont s'afficher.

Nous avons rajouté une option ***--help*** pour que l'utilisateur puisse avoir des informations sur la commande.

Les limites de la protection par adresses MAC

Le filtrage MAC n'est pas une solution de sécurité aussi performante qu'auparavant, certains pirates parviennent à la contourner facilement : grâce à des logiciels spécialisés (appelés *sniffers*), ils peuvent « écouter » à distance les informations qui circulent sur un réseau Wi-Fi, jusqu'à repérer une ou plusieurs adresses MAC acceptées sur le réseau. Bien que cela soit illégal, les pirates peuvent ensuite modifier l'adresse MAC de leur ordinateur portable, et la faire correspondre avec une adresse acceptée sur le réseau visé. Cette opération peut être assez rapide. Le filtrage MAC reste cependant une sécurité complémentaire qui peut repousser certaines intrusions. Il serait par contre risqué de miser toute la sécurité de son réseau sur ce seul dispositif.

Le code :

```
#!/bin/bash

if [ "$#" == 0 ]; then
echo "aucun parametre, pour plus d'information --help "
exit
fi

if [ "$1" = "--help" ]; then
echo "     La commande s'execute en root"
echo "     La commande nmap doit etre installé"
echo "     La commande permet de trouver les adresses MAC connectées
au réseau"
echo "     Et de les comparer avec les adresses MAC supposées
connectées"
echo "     1er parametre : fichier d'adresse MAC (peut comporter des
commentairs #) "
echo "     [option] -v  verbose "
exit
fi

if [ ! -f $1 ]; then
echo "$1 n'est pas un fichier, plus d'information --help"
exit
fi

#on recupér l'ip
IP=` ifconfig | awk 'NR == 2 {print;}' | awk '{  print $2 }' | sed
's/\.[0-9]*$/\.\*/' `

#on recuper les adresse mac connecté au reseau
declare -a MAC=(`nmap -sP "$IP" | grep MAC | awk '{ print $3}' | sort
`)

#on lit dans le fichier les adresses mac que l'on connait
#on fait attention à supprimer les espaces et les lignes qui ne
contiennent pas d'adresses Mac
#espace ou ligne en commentaire commencent par #
declare -a MACKNOW=(`sed -e '/^[ ]*$/d' -e '/#/d' $1 | sort`)

IP=` ifconfig | awk 'NR == 2 {print;}' | awk '{  print $2 }' `
echo "votre adresse ip : $IP"
```

```
if [ "$2" == "-v" ]; then
echo "Adresse mac connecté au réseau :"
for t in "${MAC[@]}"
do
echo $t
done

echo "Adresse mac connecté au autorisé :"
for tk in "${MACKNOW[@]}"
do
echo $tk
done
fi


RETURN=0

NB=${#MACKNOW[@]}

#on compart les 2 tableau
for R in ${MAC[@]} ; do

BOOL_ADRESS=0

#on regarde si R est dans MACKNOW
for (( I=0 ; I<$NB ; I++ )); do
if [ $R == ${MACKNOW[$I]} ]; then
BOOL_ADRESS=1
if [ $# -gt 1 ] && [ $2 == "-v" ];then
echo "OK pour $R"
fi
fi
done

if [ $BOOL_ADRESS == 0 ]; then
echo "attention inconnu : $R"
RETURN=1
fi

done

if [ $RETURN == 0 ]; then
echo "Il n'y a aucun intrus"
fi
```