Atelier: Espionner une conversation telnet

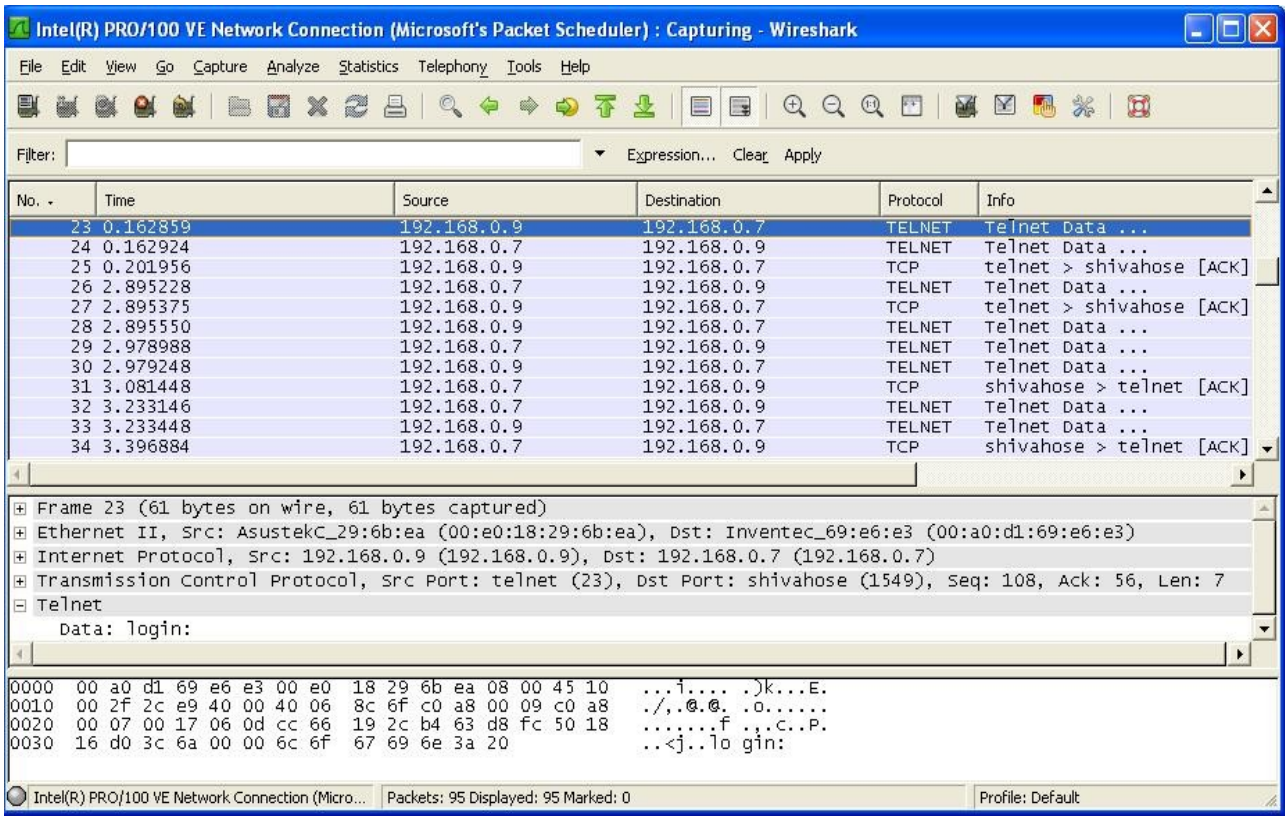
Les adresses ip présentées dans cet atelier ne sont pas nécessairement les mêmes que celles manipulées au laboratoire …

- Lancer *Wireshark* sur le poste client.
- Définir un filtre de capture pour telnet *(tcp.port==23)* et pour l'interface concernée.
- Lancer la capture.
- Etablir une connexion telnet.

```
# telnet @IP_Serveur_Telnet

 Login: XXXXX
 Password: yyyyyyy
```

- Analyser le contenu des trames en vue d'y retrouver les identifiants.

  Ici, chaque lettre du login et du mot de passe est contenue dans une trame.
  Soit les identifiants suivants: `gouwy/azerty`



Ici, c'est la trame 23 qui contient l'envoi de l'invite 'login:' du serveur vers le client.
En ouvrant les trames suivantes, vous découvrirez lettre/lettre le login de connexion.
Il en va de même, avec le mot de passe dont l'invite est envoyé ici en trame 44....
Toute la communication a lieu en clair sur le réseau !!!!

Ateliers hachage:

a. Utilisation de la commande MD5

Générez une empreinte sur un fichier

```
# cat > f1.txt
Bonjour, c'est moi !
# md5sum f1.txt > f1.txt.md5
# cat f1.txt.md5
90ffc18c8c4c8fc4f1412b855f14d790  f1.txt
```

➔ *Le condensat contient aussi le nom du fichier*

Vérification de cette empreinte sur:

a) Ce même fichier non modifié:

```
# md5sum -c f1.txt.md5
```
→ *Pas besoin de rappeler le nom du fichier car il est dans l'empreinte.*

```
f1.txt: OK
# echo $?
0
```

b) Ce même fichier modifié:

```
# echo ! >> f1.txt
# md5sum -c f1.txt.md5
f1.txt: ECHEC
md5sum: AVERTISSEMENT : 1 de 1 somme de contrôle ne concorde pas.
# echo $?
1
```

b. Vérification de l'intégrité d'un logiciel téléchargé

Vérifiez l'intégrité du package webmin (http://www.webmin.com)

1. Téléchargez Webmin (version minimale)

```
# wget http://prdownloads.sourceforge.net/
                          webadmin/webmin-1.670-minimal.tar.gz
...
Sauvegarde en : «webmin-1.670-minimal.tar.gz»

100%[======================>] 2 587 726   3,46M/s   ds 0,7s

2014-03-03 12:52:39 (3,46 MB/s) - «webmin-1.670-
minimal.tar.gz» sauvegardé [2587726/2587726]
```

2. Vérification de son empreinte MD5

```
# md5sum webmin-1.670-minimal.tar.gz
f6a99c0ec6a805e76235ed991ec474a8  webmin-1.670-minimal.tar.gz
```

La signature est correcte car identique à celle présentée sur le site de téléchargement.

MD5 Verification

To verify that you have downloaded Webmin fully and correctly, you can use the command md5sum on the RPM, Debian package or TAR file, and compare it against those listed below :

Filename	MD5 Checksum
webmin_1.670_all.deb	e1191f1c263c01cc4b33ff3853068fd4
webmin-1.670-minimal.tar.gz	f6a99c0ec6a805e76235ed991ec474a8
webmin-1.670-1.noarch.rpm	a67405a7d39407d4f47dd20883b01883
webmin-1.670-1.src.rpm	e20902361cd41847845df20c5c94536f
webmin-1.670.pkg.gz	13db948a3c592bb2173135600fd857b5
webmin-1.670.tar.gz	db2edf10ad1eb0cc8d80849a61dafd3a
webmin-1.670.zip	e821eb4d431bef06ecc091f71b551b80

Atelier: Espionner une conversation ssh

Les adresses ip présentées dans cet atelier ne sont pas nécessairement les mêmes que celles manipulées au laboratoire …

- Lancer *Wireshark* sur le poste client.

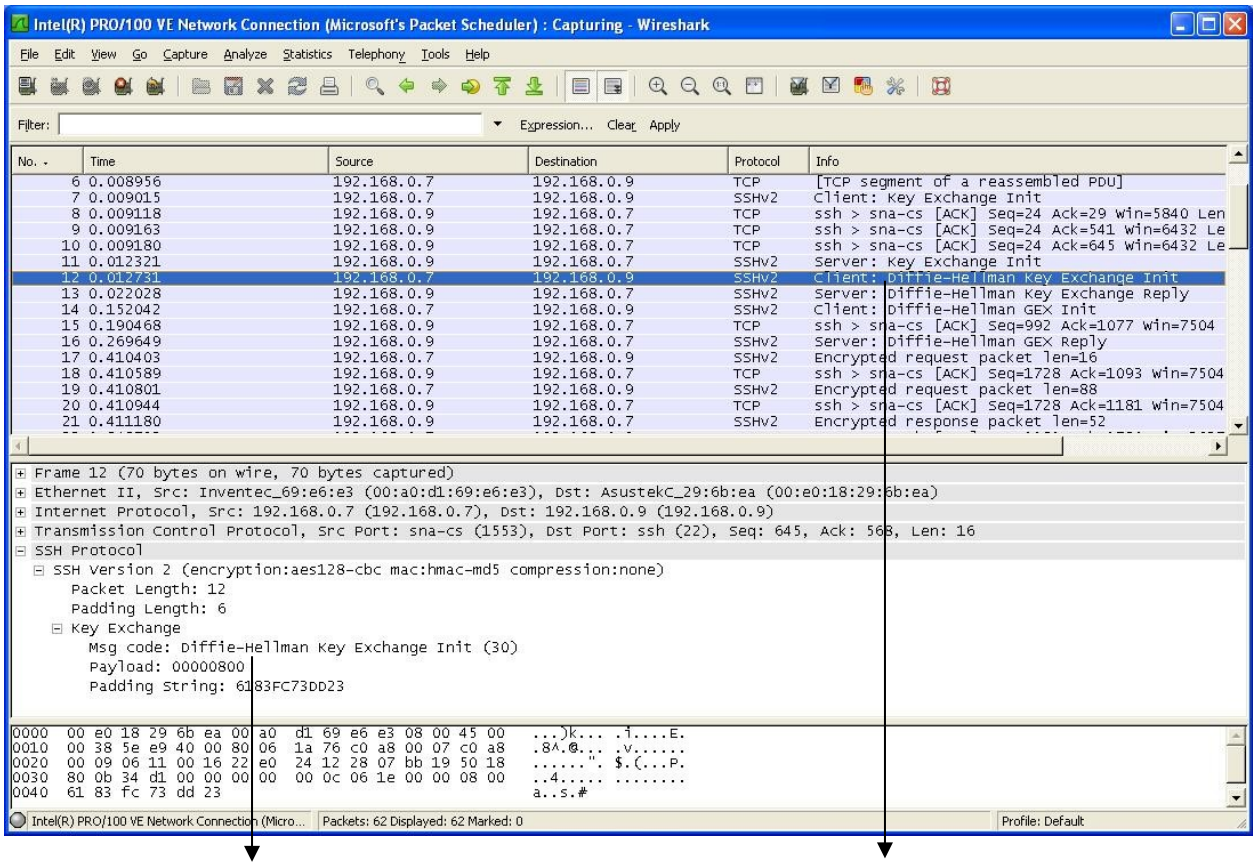
- Définir un filtre de capture pour ssh *(tcp.port==22)* et pour l'interface concernée.

- Lancer la capture.

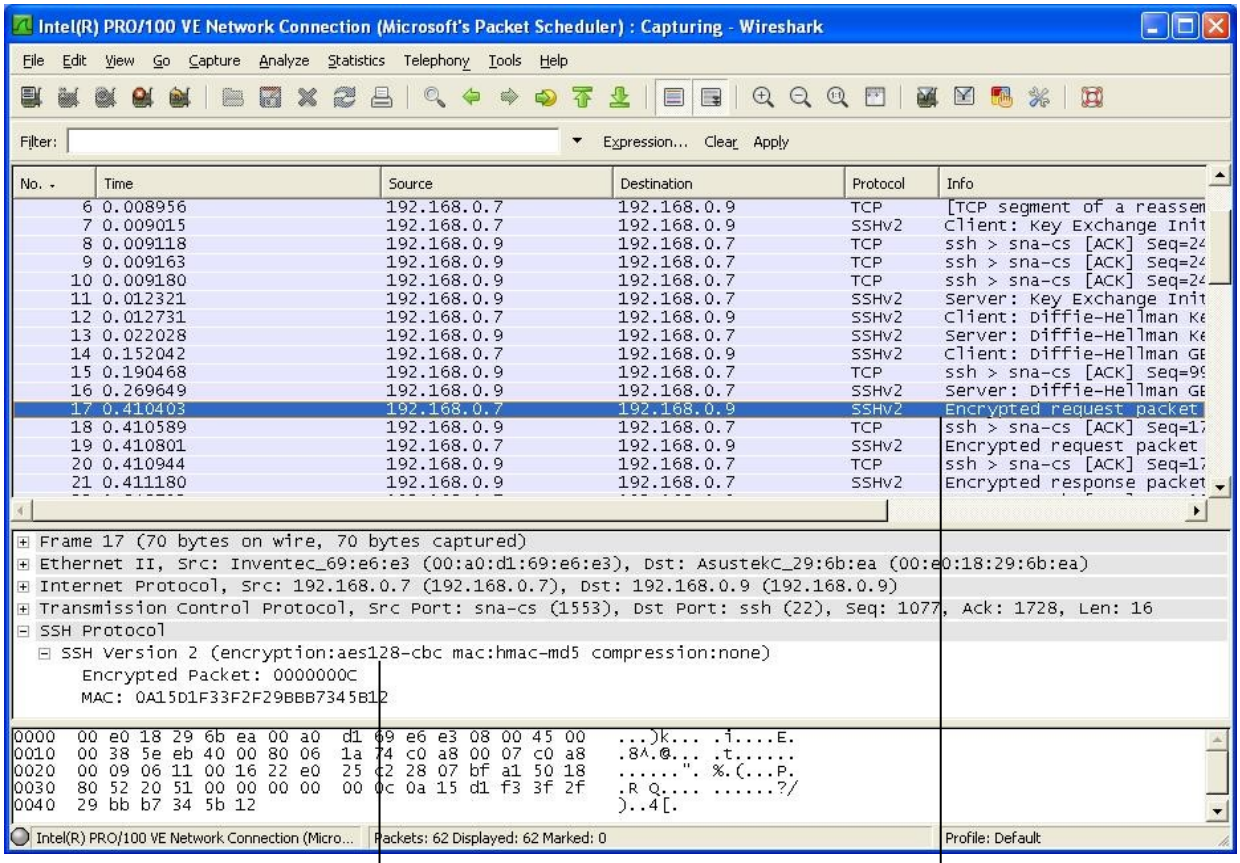
- Etablir une connexion ssh.

```
# ssh @IP_Serveur_ssh

Login: XXXXX
Password: yyyyyyy

$
```

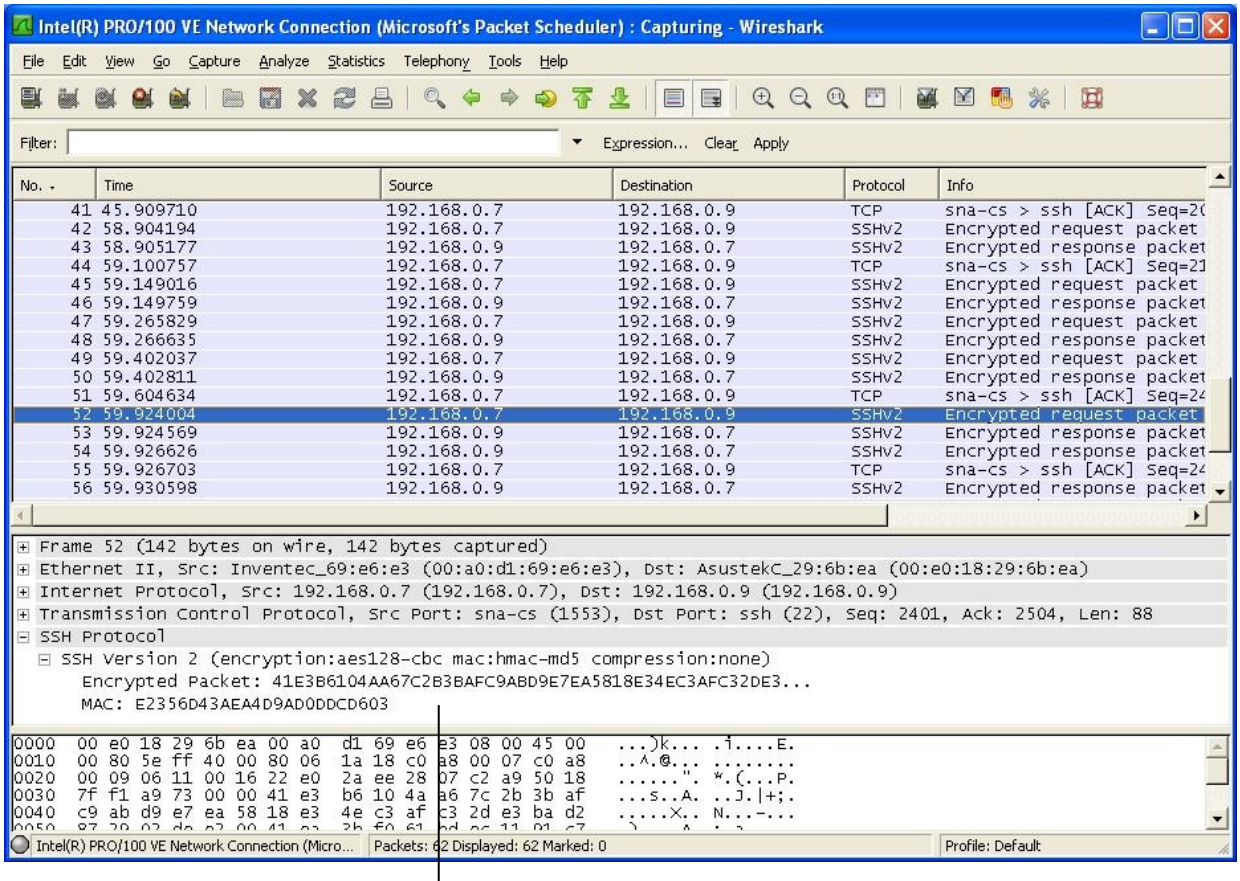
- Constatations: Négociation des algorithmes, Diffie-Hellman, chiffrement des paquets ...



Ici, de la trame 12 à la trame 16, on constate que la clé de session est négociée en Diffie-Helman.

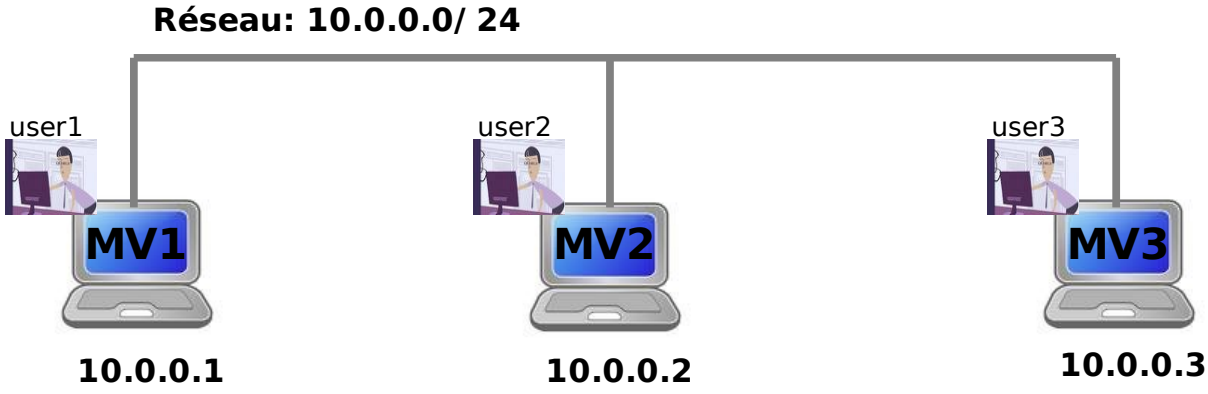


A partir de la trame 17, tous les paquets client vers serveur (ou inversément) sont chiffrés symétriquement à l'aide de l'algorithme AES



A partir de ce moment , toute la suite de la communication est chiffrée ...

Exercice: Connexion par mot de passe



Chaque machine doit accepter des connexions par mot de passe.

Pour ce faire:

- vérifiez qu'aucun coupe-feu ne tourne sur aucune machine

Sur chaque station:

```
# iptables –L
# service iptables stop
# chkconfig --level 35 iptables off
```

- vérifiez sur chaque machine qu'un trousseau de clés existe

Sur chaque station:

```
# ls –l /etc/ssh
```

Si les trousseaux sont pareils cela provient du clonage des machines.
On va donc regénérer sur MV2 & MV3 des nouveaux trousseaux de clés.

Sur MV2 & MV3:

. Supprimer toutes les clés du serveur

. # service sshd restart
afin qu'il regénère ses trousseaux

ou

```
. # ssh-keygen –t dsa –N’’ –f /etc/ssh/ssh_host_dsa_key
  # ssh-keygen –t dsa –N’’ –f /etc/ssh/ssh_host_rsa_key
```

- configurez un service sshd sur chaque machine et lancez le service (bien vérifiez que la directive PasswordAuthentication est à yes)

Sur chaque station:

```
# cat /etc/ssh/sshd_config | grep Password
# service sshd restart
# ps ax | grep sshd
```

- créez user1 sur MV1, user2 sur MV2 et user3 sur MV3

```
root@MV1# adduser user1
root@MV1# passwd user1

root@MV2# adduser user2
root@MV2# passwd user2

root@MV3# adduser user3
root@MV3# passwd user3
```

- testez les connexions suivantes:

```
user1@MV1$ slogin user2@10.0.0.2
user1@MV1$ slogin user3@10.0.0.3

user2@MV2$ slogin user1@10.0.0.1
user2@MV2$ slogin user3@10.0.0.3

user3@MV3$ slogin user2@10.0.0.2
user3@MV3$ slogin user1@10.0.0.1
```

- visualisez les ports d'écoute (client/serveur) à travers une connexion

Shell 1 sur MV1: `root@MV1# slogin user2@10.0.0.2`

Shell 2 sur MV1: `root@MV1# netstat -tn`

- utilisez *ssh* et *scp* (après avoir consulter les pages de manuel ☺)

```
root@MV1# ssh user2@10.0.0.2 'cat /etc/passwd'
```

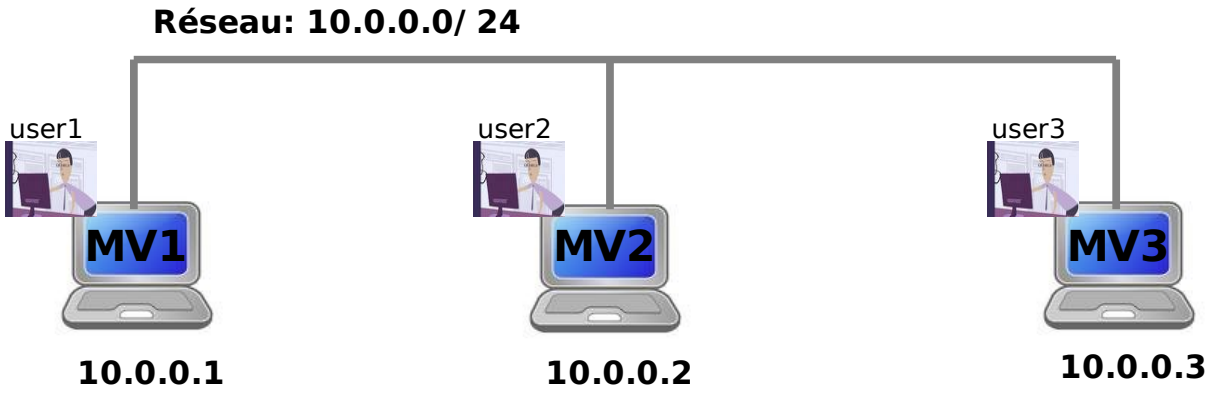
```
root@MV1# scp /root/install.log user2@10.0.0.2:/tmp
```

(Le propriétaire devient user2)

- explorez les fichiers `/etc/ssh/*` et `~/.ssh/known_hosts`

Voir laboratoire

Exercice: Connexion par clés



Faites en sorte que *user1* puisse se connecter sur MV3 sous *user3* sans entrer de mot de passe. Pour ce faire:

- `root@MV3#` → vérifie des clauses `RSAAuthentication` et `PubkeyAuthentication` à `yes` dans `/etc/ssh/sshd_config`

```
root@MV3# cat /etc/ssh/sshd_config | grep Authen
```

- `user1@MV1$` → génère un trousseau de clés de type RSA (sans passphrase)

```
user1@MV1$ ssh-keygen –t rsa
```

- `user1@MV1$` → copie la clé publique de `user1` dans un fichier `authorized_keys` de ce dossier (attention aux permissions)

```
user1@MV1 .ssh$ ssh-copy-id –i id_rsa.pub user3@10.0.0.3
```

Permissions	
`/home/user3`	→ 700
`/home/user3/.ssh`	→ 700
`/home/user3/.ssh/authorized_keys`	→ 600

- user1@MV1$ → se connecte sous user3 sur MV3.

```
user1@MV1$ ssh user3@10.0.0.3
```

Explorez les fichiers ~/.ssh/* et ~/.ssh/authorized_keys

Voir laboratoire

Exercices: Gestion d'un parc Linux

/root/admin/initvar.sh

```
#! /bin/bash
PUBKEYDIR="/root/.ssh"
SHDIR="/root/admin/sh"
LOGDIR="/root/admin/log"
IPFILE="/root/admin/ip.txt"
NAMELOG=`basename $0`
```

/root/admin/ip.txt

```
10.0.0.2
10.0.0.3
```

Exercice 1:

Ecrivez et testez un script (pushkey.sh)qui déploie la clé publique de root de la machine d'administration (MV1) vers toutes les stations du parc.

/root/admin/sh/pushkey.sh

```
#! /bin/bash

. /root/admin/initvar.sh

cd $SHDIR
date > $LOGDIR/$NAMELOG.log
date > $LOGDIR/$NAMELOG.errors.log

for IP in `cat $IPFILE`
do
  if ping -c 2 $IP >/dev/null 2>&1
  then
     ssh-copy-id -i $PUBKEYDIR/id_rsa.pub root@$IP
     echo "Copie vers $IP... OK" >> $LOGDIR/$NAMELOG.log
  else
     echo "$0: $IP ne repond pas" >> $LOGDIR/$NAMELOG.errors.log
  fi
done
exit 0
```

Exercice 2:

Ecrivez et testez un script (`haltall.sh`)qui éteint toutes les stations du parc encore « on-line ».

Programmez l'exécution de ce script à 21h00 tous les jours.
Pour ce faire le package `crontab` doit être installé...

```
# ps ax | grep crond
# yum install crontab (si nécessaire)

# crontab –e
00 21 * * * sh /root/admin/sh/haltall.sh
```

/root/admin/sh/haltall.sh

```
#! /bin/bash

. /root/admin/initvar.sh

cd $SHDIR
date > $LOGDIR/$NAMELOG.log
date > $LOGDIR/$NAMELOG.errors.log

for IP in `cat $IPFILE`
do
  if ping -c 2 $IP >/dev/null 2>&1
  then
     ssh root@$IP "shutdown -h now"
     echo "Arret de $IP... OK" >> $LOGDIR/$NAMELOG.log
  else
     echo "$0: $IP ne repond pas" >> $LOGDIR/$NAMELOG.errors.log
  fi
done
exit 0
```

Exercice 3:

Ecrivez et testez un script (chpwdroot.sh)qui change mot de passe de root sur toutes les stations du parc. Le nouveau de passe est passé en argument au script.

/root/admin/sh/chpwdroot.sh

```
#! /bin/bash

. /root/admin/initvar.sh

if [ $# -eq 1 ]
then
  cd $SHDIR
  date > $LOGDIR/$NAMELOG.log
  date > $LOGDIR/$NAMELOG.errors.log

  for IP in `cat $IPFILE`
  do
    if ping -c 2 $IP >/dev/null 2>&1
    then
       ssh root@$IP "echo $1 | passwd --stdin root > /dev/null
                                                      2>&1"
       echo "Changement du mdp de root sur $IP... OK"
                                          >> $LOGDIR/$NAMELOG.log
    else
       echo "$0: $IP ne repond pas"
                                    >> $LOGDIR/$NAMELOG.errors.log
    fi
  done
else
  echo "Erreur: Un et un seul argument"
  exit 1
fi
exit 0
```

Exercice 4:

Ecrivez et testez un script (`alladduser.sh`)qui ajoute un compte utilisateur à chaque station du parc.

Le nom de l'utilisateur (ex. `toto`) sera passé en argument. Son mot de passe sera identique à son nom.

/root/admin/sh/alladduser.sh

```
#! /bin/bash

. /root/admin/initvar.sh

if [ $# -eq 1 ]
then
  echo $LOGDIR/$NAMELOG
  cd $SHDIR
  date > $LOGDIR/$NAMELOG.log
  date > $LOGDIR/$NAMELOG.errors.log

  for IP in `cat $IPFILE`
  do
    if ping -c 2 $IP >/dev/null 2>&1
    then
       ssh root@$IP "adduser $1 > /dev/null 2>&1"
       ssh root@$IP "echo $1 | passwd --stdin $1 > /dev/null
                                                    2>&1"
       echo "Ajout de l'utilisateur $1 sur $IP... OK"
                                        >> $LOGDIR/$NAMELOG.log
    else
       echo "$0: $IP ne repond pas"
                                  >> $LOGDIR/$NAMELOG.errors.log
    fi
  done
else
  echo "Erreur: Un et un seul argument"
  exit 1
fi
exit 0
```

Exercice 5:

Ecrivez et testez un script (chgrub.sh) qui permet de changer le 'timeout' du multi-boot de chaque machine à 5 secondes.

/root/admin/sh/chgrub.sh

```
#! /bin/bash

. /root/admin/initvar.sh

cd $SHDIR
date > $LOGDIR/$NAMELOG.log
date > $LOGDIR/$NAMELOG.errors.log

for IP in `cat $IPFILE`
do
  if ping -c 2 $IP >/dev/null 2>&1
  then
     ssh root@$IP "/bin/sed –i –e \"s/timeout=5/timeout=0/g\"
                                          /boot/grub/grub.conf"
     echo "Change timeout on $IP... OK" >> $LOGDIR/$NAMELOG.log
  else
     echo "$0: $IP ne repond pas" >> $LOGDIR/$NAMELOG.errors.log
  fi
done
exit 0
```

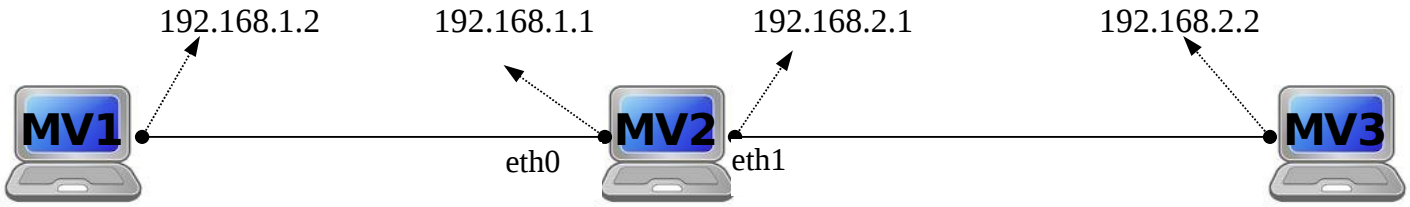
Exercice 6:

Ecrivez et testez un script (chnetwork.sh)qui change la configuration ip de chaque station du parc afin de les disposer sur le réseau d'adresse 192.168.0.0/24.

/root/admin/sh/chnetwork.sh

```
#! /bin/bash

. /root/admin/initvar.sh

  cd $SHDIR
  date > $LOGDIR/$NAMELOG.log
  date > $LOGDIR/$NAMELOG.errors.log

  cpt=2
  for IP in `cat $IPFILE`
  do
    if ping -c 2 $IP >/dev/null 2>&1
    then
       echo "DEVICE=eth0" > /tmp/ifcfg-eth0
       echo "BOOTPROTO=static" >> /tmp/ifcfg-eth0
       echo "IPADDR=192.168.0.$cpt" >> /tmp/ifcfg-eth0
       echo "NETMASK=255.255.255.0" >> /tmp/ifcfg-eth0
       echo "ONBOOT=yes" >> /tmp/ifcfg-eth0
       ssh root@$IP "cp /etc/sysconfig/network-scripts/
                                 ifcfg-eth0 /root/ifcfg-eth0.bak"
       scp /tmp/ifcfg-eth0 root@$IP:/etc/sysconfig/network-
                               scripts/ifcfg-eth0 > /dev/null 2>&1
       echo "Changement de la configuration de $IP en
                    192.168.0.$cpt... OK" >> $LOGDIR/$NAMELOG.log
       cpt=`expr $cpt + 1`
    else
       echo "$0: $IP ne repond pas"
                                    >> $LOGDIR/$NAMELOG.errors.log
    fi
  done
rm -f /tmp/ifcfg-eth0
exit 0
```

Exercice: Le port forwarding



Réalisez la maquette présentée ci-dessus (virtualisation: réseau interne)

- vérifiez qu'aucun coupe-feu ne tourne sur aucune machine

Sur chaque station:

```
# iptables –L
# service iptables stop
# chkconfig --level 35 iptables off
```

- n'oubliez pas d'activer l'ip forwarding sur MV2

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

- stopper les services sshd sur MV1 et MV2 (s'ils tournent)

```
MV1# service sshd stop
MV2# service sshd stop
```

- configurez un service sshd sur MV3 et relancez le service (bien vérifiez que la directive 'PasswordAuthentication' est à yes)

```
# cat /etc/ssh/sshd_config | grep Password
# service sshd restart
# ps ax | grep sshd
```

- vérifiez sur MV3 qu'un trousseau de clés existe

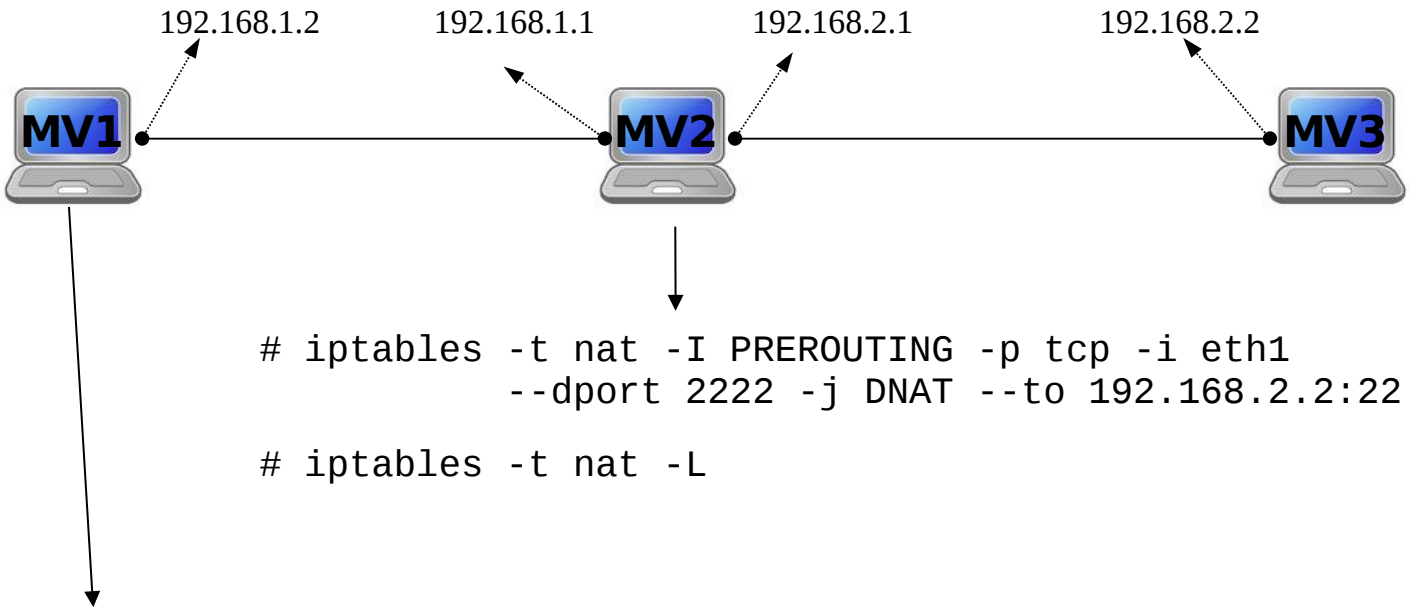
```
MV3# ls –l /etc/ssh
```

- activez le forwarding de port sur MV2 et tentez de joindre MV3 à partir de MV1 par mot de passe.

- créez user3 sur MV3

```
root@MV3# adduser user3
root@MV3# passwd user3
```

Configurez et testez un forwarding de port de MV1 vers MV3 en passant par MV2.




```
# iptables -t nat -I PREROUTING -p tcp -i eth1
           --dport 2222 -j DNAT --to 192.168.2.2:22

# iptables -t nat -L
```

```
# slogin -p 2222 user3@192.168.1.1
Password: …
```

ou

```
# ssh -p 2222 –l userb 192.168.1.1
Password: …
```

Atelier: Tunneling

Les adresses ip présentées dans cet atelier ne sont pas nécessairement les mêmes que celles manipulées au laboratoire …

Sniffing d'une session Ftp non tunnelisée

Serveur (192.168.0.9) – (proftpd / sshd)

```
# tshark –V –i eth0 -R ftp | tee ftpsniff.txt
…
…
Request command: USER
Request arg: ftpuser      → le login apparaît en clair.
…
…
Request command: PASS
Request arg: ftpuser      → le mot de passe apparaît en
clair.
```

Client Texte (cmde ftp)

```
# ftp 192.168.0.9
…
…
Name: ftpuser
Password: ftpuser
…                          } Session Ftp
…
ftp> quit
#
```

Client graphique Linux (gftp)

```
# gftp
<<< Apparition d'une fenêtre graphique >>>
Hôte:  192.168.0.9 Port:
Utilisateur:  ftpuser
Mot de passe:  ftpuser
FTP
```

Client graphique Filezilla

```
Adresse:  192.168.0.9  Utilisateur:  ftpuser
Mot de passe: ftpuser  Port:  21  Connexion rapide
```

Conclusion:

Toute la communication Ftp passe en clair sur le réseau !

Sniffing d'une session Ftp non tunnelisée

Serveur (192.168.0.9) – (proftpd / sshd)

```
# tshark –V –i eth0 -R ftp | tee ftpsniff.txt

<<< Aucun affichage car ce qui entre par eth0 n'est pas destiné au port ftp
      mais au port ssh du serveur >>>

<<< Sniffons alors le port ssh (22)  >>>

# tshark –V –i eth0 –R ssh | tee ftpsniff.txt
…
…
…
<<< On y découvre plus jamais les chaînes 'USER', 'PASS' et 'ftpuser' …>>>
```

Client Texte Linux (la commande ftp)

```
# ssh –N –f -L 2121:192.168.0.9:21 ftpuser@192.168.0.9
# ftp localhost 2121
…                                          Canal des commandes.
…
Name: ftpuser
Password: ftpuser
…                    }
…                    } Session Ftp
ftp> quit            .
#
```

<u>Conclusion</u>:

Toute la communication Ftp passe cryptée sur le réseau !