Compte rendu TP2

Connexion aux machines en SSH

Pour ce TP nous avons réutilisé nos machines créées lors du TP1. Cette fois ci il ne fallait pas se connecter directement sur les machines mais se connecter en SSH. Les trois machines se trouvaient dans un réseau NAT et ne pouvaient donc pas communiquer avec l'hôte. Pour rendre cela possible il fallait procéder à un forwarding de port.

Méthode : dans le menu de VirtualBox cliquer sur Fichier > Paramètres > Réseau > RT0701_TP1 (que nous avions crée dans le TP1) > Redirection de ports.

Il suffit ensuite de compléter le tableau avec les informations suivantes :

- Nom : nom qui permet de décrire la redirection
- Protocole : TCP
- IP hôte : ajouter l'adresse IP localhost 127.0.0.1. Pas besoin de mettre celle de la carte réseau de l'hôte quand on ne souhaite pas de connexions extérieures
- Port hôte : choix du numéro de port pour se connecter en SSH
- IP invité : adresse IP de la machine virtuelle
- Port invité : 22 pour se connecter en SSH.

J'avais donc à la fin ce fichier de configurations :

Nom	Protocole	IP hôte	Port hôte	IP invité	Port invité
SSH Alpine	TCP	127.0.0.1	2221	172.18.10.11	22
SSH Debian	TCP	127.0.0.1	2220	172.18.10.10	22
SSH Ubuntu	TCP	127.0.0.1	2222	172.18.10.12	22

Etant sous Windows je me suis ensuite connectée en SSH aux machines, ici Ubuntu, par l'invite de commande suivante :

```
ssh superv@127.0.0.1 -p 2222
```

Il me suffisait ainsi de changer le numéro de port pour me connecter respectivement à la machine Alpine, Ubuntu et Debian.

Gestion des logs

Sur chacune des machines l'ensemble des opérations d'authentification (locale, SSH, su, sudo) seront loggées. En effet, toutes les opérations réussies seront enregistrées en local sur chaque machine dans le fichier /var/log/opOK.log.

Dans un premier temps, j'ai téléchargé Syslog-ng :

```
apt-get update
apt-get install syslog-ng
```

Dans le fichier de configuration : `/etc/syslog-ng/syslog-ng.conf` j'ai trouvé une ligne indiquant de créer nos fichiers de configurations dans le dossier `/etc/syslog-ng/conf.d/` :

```
custom configuration can be added under this folder
/etc/syslog-ng/conf.d/*.conf
```

J'ai donc créé un fichier de configuration pour y ajouter mes règles :

`nano /etc/syslog-ng/conf.d/send_to_debian.conf`

Les informations sur les connections sont stockées dans le fichier : `/var/log/auth.log`. Il fallait donc créer une règle source qui vienne récupérer les informations dans ce fichier

```
source auth_src {
      file("/var/log/auth.log");
};
```

Puis, j'ai rajouté la règle filtre pour dire que je souhaitais garder seulement les logs de succès :

```
filter r1_filt{match("authentication failure" value("MESSAGE"))};
filter r2_filt{match("Failed" value("MESSAGE"))};

filter auth_failed_filt{
        filter(r2_filt)
or
        filter(r1_filt);
};

filter auth_filt{
        not filter(auth_failed_filt);
};
```

⚠ Ne pas oublier le paramètre « `value` ».

J'ai ensuite créé la règle destination pour envoyer les logs choisis dans le fichier « `opOK.log` » comme le demande la consigne :

```
destination auth_dest {
      file("/var/log/opOK.log");
};
```

Enfin j'ai créé le lien entre ces règles :

```
log {
      source(auth_src);
      filter(auth_filt);
      destination(auth_dest);
};
```

J'ai redémarré Syslog-ng pour qu'il prenne en compte mes modifications avec la commande :
`/etc/init.d/syslog-ng restart`

Toutes les erreurs seront quant à elles loggées sur le serveur Syslog-ng de la machine debian dans différents fichiers. L'architecture des fichiers permettra de repérer la machine et le type d'accès illégal.

- Configuration côté serveur, Debian :

Dans un premier temps j'ai créé le fichier de configuration côté serveur (Debian) :

`nano /etc/syslog-ng/conf.d/error_log_server.conf`

J'ai rajouté une règle source pour qu'il écoute et j'ai rajouté des options pour être sûr qu'il ait les permissions :

```
options{
        create_dirs(yes);
```

```
        owner(root);
        group(root);
        perm(0640);
        dir_owner(root);
        dir_group(root);
        dir_perm(0750);
};
source ubuntu_src{
        udp(ip(0.0.0.0) port(51401));
        tcp(ip(0.0.0.0) port(51401));
};
```

J'ai ensuite rajouté la règle destination ainsi que le log final :

```
destination ubuntu_dest{
        file("/var/log/erreur_sur_ubuntu/$HOST.log");
};

log{
        source(ubuntu_src);
        destination(ubuntu_dest);
};
```

Puis j'ai redémarré le service Syslog avec la commande : `service syslog-ng start`

- Configuration côté client, Ubuntu :

J'ai complété le fichier de configuration créé plus tôt. J'ai donc inséré les règles de destination en lui donnant l'adresse IP du serveur, donc celle de la machine Debian :

```
destination debian_dest{
        tcp("172.18.10.10" port(51401) localport(999));
};
```

J'avais déjà créé les règles de filtres, pour récupérer les logs de succès, et de source, j'ai donc seulement récupéré ces règles.

```
log {
      source(auth_src2);
      filter(auth_failed_filt);
      destination(debian_dest);
};
```

J'ai bien pensé à redémarrer Syslog-ng à la fin de mes paramétrages.

Bilan : commandes utiles pour syslog-ng

Permet de lister l'état des connexions / sockets :
```
apt-get install net-tools
netstat
```

Commande qui permet de débuguer :
```
service syslog-ng -d
```

Commande qui permet de regarder les derniers logs :
```
tail -f /var/log/syslog
```

Attention version

⚠ Au départ, le fichier de configuration de syslog-ng n'était pas compatible avec la version installée. J'ai dû me rendre dans « `/etc/syslog-ng/syslog-ng.conf` » et changer le numéro de version. La commande « `service syslog-ng -d` » m'avait permis de repérer cette erreur.

Supervision basée sur SNMP

Installation des paquets nécessaires en sachant que le NMS sera la machine Ubuntu

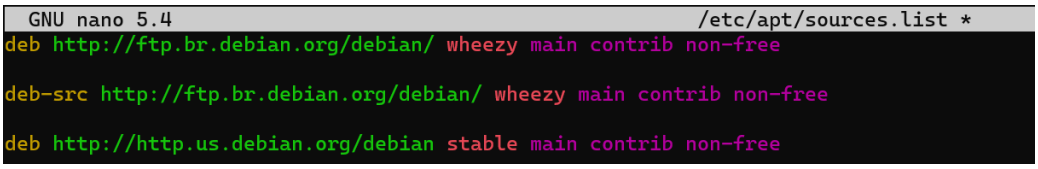
Sur la machine Ubuntu, le NMS

J'ai téléchargé les paquets SNMP et vérifié que la ligne « `mibs` » n'était pas commentée dans le fichier de configurations :

```
sudo apt-get update
sudo apt-get install snmp
sudo nano /etc/snmp/snmp.conf
```

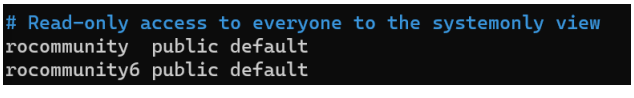
Sur la machine Debian

Les configurations ont été un peu plus longues. En effet, après avoir téléchargé les paquets SNMPD avec la commande « `sudo apt-get install snmpd` » il fallait télécharger les librairies MIBS qui permettent de pouvoir lire les nœuds et donc de récupérer des informations sur le système de la VM. Le problème était que le paquet « `snmp-mibs-downloader` » est « non-free » ce qui implique qu'il faut rajouter dans le fichier « `/etc/apt/sources.list` » les termes « `contrib non-free` » à la fin de chaque ligne (cf. image). C'est une fois ces modifications faites que l'installation du paquet peut aboutir. Voici donc les commandes nécessaires que j'ai réalisé :




```
sudo apt-get update
sudo apt-get install snmpd
sudo nano /etc/apt/sources.list
#ajouter 'contrib non-free'
sudo apt-get update
apt-get install snmp-mibs-downloader
```

Il fallait ensuite réaliser les configurations. J'ai dans un premier temps créé une copie du fichier de configuration pour pouvoir le restaurer en cas de problèmes. Puis, J'ai mis en commentaire la ligne « `mibs` » du fichier de configuration « `/etc/snmp/snmp.conf` ». Toujours dans le même fichier, il faut penser à supprimer « `-V systemonly` » derrière les lignes « `recommunity` », cf. image, ainsi qu'à changer la valeur de «`agentaddress` ». Il faut remplacer la local host par « `udp:161` ». Ce qui donnait :




```
cp /etc/snmp/snmpd.conf /etc/snmp/snmpd.conf.bak
nano /etc/snmp/snmpd.conf
#commenter 'mibs'
#enlever les '-V systemonly'
agentaddress udp:161
```

Enfin, j'ai pensé à relancer les services et à vérifier mes configurations :

```
systemctl status snmpd
sudo service snmpd restart
netstat -nlpu | grep snmp
```

Configuration des machines afin de pouvoir récupérer les informations de charge mémoire, cpu et d'occupation de disque par une requête SNMP

Afin de récupérer les diverses informations j'ai utilisé …

Pour …	*La commande …*
La charge mémoire	`snmpwalk -v 2C -c public 172.18.10.10 1.3.6.1.4.1.2021.4.6.0`
Le CPU	`snmpwalk -v 2C -c public 172.18.10.10 .1.3.6.1.4.1.2021.11.10.0`
L'occupation de disque	Normalement le nœud permettant de récupérer l'occupation du disque se trouve dans « `.1.3.6.1.4.1.2021.9`ais il ne fonctionnait pas. Donc je l'ai rajouté à la main, cf. la partie méthode « Utiliser Extend pour créer des OID» ci-dessous.

A noter que les nœuds ont été trouvé ici : http://www.debianadmin.com/linux-snmp-oids-for-cpumemory-and-disk-statistics.html

Utiliser Extend pour créer des OID

Cette méthode utilise comme exemple la récupération de l'occupation du disque.

Dans un premier temps j'ai créé un script qui contient la commande que je souhaitais exécuter. Ici la commande me permet de récupérer l'espace de libre dans le disque. Puis, je rajoute les droits d'exécution au script.

```
nano used_disk_script.sh
#!/bin/bash
df -h | awk /sda/ | awk '{print $5}'
^X
chmod +x used_disk_script.sh
```

J'ai ensuite rajouté un « `extend` » à la fin du fichier de configuration « `/etc/snmp/snmpd.conf` ». Pour que les mises à jour soient prises en compte j'ai redémarré le service.

```
chmod +x used_disk_script.sh
nano /etc/snmp/snmpd.conf
extend used_disk "/home/superv/used_disk_script.sh"
^X
sudo service snmpd restart
```

J'ai récupéré le nom de mon nœud avec la commande « `snmpwalk -v 2C -c public 127.0.0.1 nsExtendOutput1` ». Il faut récupérer la ligne qui permet de retourner les valeurs qui nous intéressent. Puis, j'ai traduit le nom de mon nœud récupéré en OID avec la commande :

```
snmptranslate -On 'NET-SNMP-EXTEND-MIB::nsExtendOutputFull."used_disk"'
```

Après avoir récupéré mon OID j'ai testé en local :

```
snmpwalk -v 2C -c public 127.0.0.1
1.3.6.1.4.1.8072.1.3.2.3.1.2.9.117.115.101.100.95.100.105.115.107
```

Enfin, j'ai testé à distance depuis la machine Ubuntu :

```
snmpwalk -v 2C -c public 172.18.10.10
1.3.6.1.4.1.8072.1.3.2.3.1.2.9.117.115.101.100.95.100.105.115.107
```

Configuration des machines afin de pouvoir récupérer la liste des paquets installés et des utilisateurs connectés sur chaque machine

Pour les explications des commandes cf. la méthode juste au-dessus.

Récupérer la liste des paquets installés

```
nano installed_pck.sh
#!/bin/bash
dpkg --list | awk '{print $2}'
^X

chmod +x installed_pck.sh

nano /etc/snmp/snmpd.conf
extend installed_pck "/home/superv/installed_pck.sh"
^X

sudo service snmpd restart
snmpwalk -v 2C -c public 127.0.0.1 nsExtendOutput1
snmptranslate -On 'NET-SNMP-EXTEND-MIB::nsExtendOutputFull."installed_pck"'
```

Sur la machine Ubuntu :

```
snmpwalk -v 2C -c public -OQas 172.18.10.10
.1.3.6.1.4.1.8072.1.3.2.3.1.2.13.105.110.115.116.97.108.108.101.100.95.112.99.107
```

On notera que l'option « `-OQas` » permet de convertir l'hexadécimal pour avoir le texte qui s'affiche correctement.

Récupérer la liste des utilisateurs connectés

Sur la machine Debian :

```
nano connected_users.sh
#!/bin/bash
w -h | awk '{print $1}'
^X

chmod +x connected_users.sh

nano /etc/snmp/snmpd.conf
extend connected_users "/home/superv/connected_users.sh"
^X

sudo service snmpd restart

snmpwalk -v 2C -c public 127.0.0.1 nsExtendOutput1
snmptranslate -On 'NET-SNMP-EXTEND-MIB::nsExtendOutputFull."connected_users"'
```

Sur la machine Ubuntu :

```
snmpwalk -v 2C -c public 172.18.10.10
1.3.6.1.4.1.8072.1.3.2.3.1.2.15.99.111.110.110.101.99.116.101.100.95.117.115.101.
114.115
```

Configuration des machines afin de pouvoir "vider" depuis le NMS les fichiers de log des accès réussis présents sur chacune des machines.

```
nano erased_logs.sh
#!/bin/bash
cd /var/log/ | rm opOK.log
echo(" #### Le fichier opOK.log a bien été supprimé. ####")
^X

chmod +x erased_logs.sh

nano /etc/snmp/snmpd.conf
extend erased_logs "/home/superv/erased_logs.sh"
^X

sudo service snmpd restart
snmpwalk -v 2C -c public 127.0.0.1 nsExtendOutput1
snmptranslate -On 'NET-SNMP-EXTEND-MIB::nsExtendOutputFull."erased_logs"'
```

Sur la machine Ubuntu

```
snmpwalk -v 2C -c public -OQas 172.18.10.10
.1.3.6.1.4.1.8072.1.3.2.3.1.2.11.101.114.97.115.101.100.95.108.111.103.115
```