R502
Supervision des réseaux
Travaux pratiques

Sami Evangelista

IUT de Villetaneuse

Département Réseaux et Télécommunications

2024–2025

http://www.lipn.univ-paris13.fr/~evangelista/cours/R502

Table des matières

TP 1 — Introduction aux outils SNMP sous Linux

Le TP est à faire en binôme sur 2 machines : une machine agent (qui est supervisée) ; et un NMS (qui supervise).

Exercice 1 — Travail préparatif

I 1.1 Restaurez une image *Debian* sur chaque poste.
I 1.2 Sur les deux postes : installez les paquets `snmp`, `snmp-mibs-downloader` et `wireshark` ; puis commentez (en rajoutant un dièse en début de ligne) la ligne `mibs :` dans le fichier `/etc/snmp/snmp.conf`.
I 1.3 Sur la machine agent : installez le paquet `snmpd` (qui installe le service homonyme).

La dernière opération permet que les commandes Net-SNMP utilisent les MIB téléchargées (après l'installation de `snmp-mibs-downloader`) et affichent les noms des objets plutôt que leurs OID (p.ex., sysName au lieu de 1.3.6.1.2.1.1.5).

Le fichier de configuration de `snmpd` (l'agent qui répond aux requêtes du NMS) est `/etc/snmp/snmpd.conf`. N'oubliez pas de redémarrer le service après chaque modification de ce fichier.

Exercice 2 — Première configuration de l'agent

I 2.1 Supprimez le fichier de configuration de l'agent (`/etc/snmp/snmpd.conf`).
I 2.2 Recréez un fichier de configuration contenant les lignes ci-dessous qui définissent le protocole de transport et le port utilisés par l'agent, et deux communautés : la première en lecture seule (`rocommunity`) et la seconde en lecture/écriture (`rwcommunity`). Remplacez `<commro>` et `<commrw>` par deux communautés de votre choix.

```
agentAddress udp:161
rocommunity <commro>
rwcommunity <commrw>
```

Q 2.1 Rappelez le rôle d'un agent SNMP.
Q 2.2 Qu'est-ce qu'une communauté SNMP ?

Exercice 3 — Commandes d'interrogation et de modification

La syntaxe générale des commandes Net-SNMP est la suivante :

```
$ snmpXXX -v <version> -c <communauté> <IP-de-l-agent> <paramètres>
```

Pour des requêtes de lecture (`XXX` = get, getnext ou bulkget), les paramètres consistent en un (ou plusieurs) OID(s).

Pour des requêtes d'écriture (`XXX` = set), on doit fournir en paramètres des triplets `<oid> <type> <valeur>`. Chaque triplet permet d'affecter une nouvelle valeur à l'objet indiqué. Le type de l'objet modifié peut prendre pour valeur : `i` (integer), `s` (string), `a` (adresse IP), … (voir le manuel de `snmpset`).

I 3.1 Sur le NMS : utilisez les commandes `snmpget`, `snmpset` et `snmpgetnext` permettant de :
(a) récupérer l'objet sysContact
(b) donner à l'objet sysContact une valeur de votre choix
(c) récupérer les objets sysDescr et sysContact
(d) récupérer l'objet suivant sysName
(e) récupérer le premier objet de la MIB (celui avec le plus petit OID)
(f) récupérer un objet permettant de savoir si la machine agent accepte de router les paquets (cherchez dans la branche IP d'oid 1.3.6.1.2.1.4, par exemple sur `http://oid-info.com`)
(g) récupérer un objet permettant de connaître le nombre de pings (messages ICMP echo) envoyés par la machine agent
(h) récupérer un objet permettant de connaître le nombre de notifications SNMP envoyées par la machine agent

Remarques :
— Une seule commande est à utiliser dans chaque cas.
— Nous utiliserons toujours SNMP version 2c.
— N'oubliez pas que pour lire ou modifier la valeur d'un objet scalaire (comme sysContact et sysDescr), il faut insérer un `.0` à la fin de son OID (ou de son nom).

Exercice 4 — Commandes de parcours de la MIB

Bien souvent, l'administrateur souhaite récupérer plusieurs objets consécutifs de la MIB d'un agent. On utilise pour cela les commandes `snmpwalk` et `snmpbulkwalk` qui ont, grosso modo, la même syntaxe que la commande `snmpget`.

I 4.1 Utilisez les commandes ci-dessous :

```
$ snmpwalk -On -v2c -c<commro> <IP-de-l-agent> 1.3.6.1.2.1.1
$ snmpbulkwalk -On -v2c -c<commro> <IP-de-l-agent> 1.3.6.1.2.1.1
```

En remplaçant `<commro>` par la communauté en lecture seule choisie à l'exercice 2 et `<IP-de-l-agent>` par l'IP de l'agent.

Q 4.1 Qu'affichent ces commandes ? (Expliquez les objets qu'elle permet de récupérer.)
Q 4.2 À quoi sert l'option `-On` ? (Cette option est valide pour toutes les commandes Net-SNMP.)

I 4.2 Relancez la commande `snmpwalk` mais, cette fois, en capturant les trames avec wireshark.

Q 4.3 En vous basant sur cette capture, expliquez le fonctionnement de la commande `snmpwalk`. Autrement dit, comment procède-t-elle pour récupérer tous les objets affichés ?

I 4.3 Relancez la commande `snmpbulkwalk` mais, cette fois, en capturant les trames avec wireshark.

Q 4.4 En vous basant sur cette capture, expliquez le fonctionnement de la commande `snmpbulkwalk`.
Q 4.5 À quoi sert l'option `-Cr` de cette commande ?

Exercice 5 — Mise en place des notifications

Nous allons maintenant configurer l'agent pour qu'il envoie des notifications (ou traps) au NMS.

Q 5.1 Rappelez l'utilité des notifications.

I 5.1 Sur l'agent : insérez les lignes ci-dessous dans le fichier de configuration de l'agent :

```
trapcommunity <communaute-de-notification>
trap2sink <IP-du-NMS>
```

La première indique la communauté utilisée pour envoyer des notifications. La deuxième ligne indique que toutes les notifications seront au format v2c (celui vu en cours) et envoyées à l'IP précisée.

Une fois cette modification effectuée, l'agent enverra automatiquement un certain nombre de notifications génériques (p.ex., suite à l'arrêt de l'agent) au NMS.

I 5.2 Sur l'agent : capturez les trames avec wireshark lors de l'arrêt puis du redémarrage du service `snmpd`. On devrait normalement capturer 2 notifications ainsi qu'un message d'erreur ICMP par lequel le NMS prévient l'agent qu'aucun processus n'écoute sur le port de destination de la notification.

Q 5.2 Donnez une copie d'écran du contenu de la notification SNMP envoyée au démarrage.
Q 5.3 Quel est le numéro de port destination utilisé par les notifications ?

Une notification, dans la version 2c de SNMP, contient toujours les deux objets suivants :
— sysUpTime (d'OID 1.3.6.1.2.1.1.3) : temps (exprimé en centièmes de secondes) écoulé depuis le démarrage de l'agent
— snmpTrapOID (d'OID 1.3.6.1.6.3.1.1.4.1) : OID du type de notification (les notifications ont un type qui est également un objet de la MIB)
On peut également y trouver d'autres objets, selon le type de notification envoyée et la configuration de l'agent.

Q 5.4 En analysant les notifications capturées, donner l'OID et le nom de la notification envoyée dans chaque cas. (Vous pouvez utiliser la commande `snmptranslate` en lui donnant en argument un OID pour qu'elle affiche le nom correspondant.)

On peut aussi configurer l'agent pour qu'il envoie des notifications en cas d'erreur de communauté.

I 5.3 En vous aidant du manuel de `snmpd.conf` (section *Notification Handling*) ou d'Internet, recherchez et ajoutez dans le fichier de configuration de l'agent la ligne permettant d'envoyer des notifications dans cette situation.

I 5.4 Tout en capturant les trames, récupérez l'objet sysContact de l'agent mais en utilisant une communauté inexistante. (La requête ne recevant pas de réponse, elle est renvoyée plusieurs fois, ce qui génère autant de notifications.)

Q 5.5 Donnez une copie d'écran du contenu de la notification envoyée.

Q 5.6 Donnez l'OID et le nom de la notification.

Exercice 6 — Extension de la MIB (bonus)

Il est possible de configurer l'agent pour qu'il exécute des commandes à la demande du NMS. Ces commandes sont "stockées" dans la branche `1.3.6.1.4.1.2021.8`, c'est-à-dire qu'à chaque fois que le NMS demande un objet de cette branche, la commande correspondante est exécutée et son résultat est envoyé au NMS sous la forme d'objets.

Utilisez la directive `exec` du fichier `snmpd.conf` (regarder le manuel ou sur Internet) pour que le NMS puisse connaître le nombre de processus sur la machine de l'agent. Écrivez pour cela un script qui écrira sur la sortie standard le nombre de processus, puis déclarez la directive `exec` dans le fichier de configuration de l'agent.

Testez en récupérant, depuis le NMS, le contenu de la branche `1.3.6.1.4.1.2021.8`.

TP 2 — Surveillance de la MIB et notifications SNMP

Ce TP est à faire en binôme sur 2 machines : une machine agent (qui est supervisée) ; et un NMS (qui supervise).

En fin de séance, vous enverrez en guise de compte-rendu :

— Les fichiers de configuration de l'agent (`/etc/snmp/snmpd.conf`) et du NMS (`/etc/snmp/snmptrapd.conf`) ;
— le script (`/bin/traitement-notification`) ;
— et un fichier PDF contenant les copies d'écran des notifications envoyés par l'agent.

Exercice 1 — Travail préparatif

I 1.1 Restaurez une image *Debian* sur les deux postes.
I 1.2 Sur les deux postes : installez les paquets `snmp`, `snmp-mibs-downloader` et `wireshark`.
I 1.3 Sur l'agent : installez le paquet `snmpd` (qui installe le service homonyme).
I 1.4 Sur le NMS : installez le paquet `snmptrapd` (qui installe le service homonyme).
I 1.5 Sur les deux postes : commentez la ligne `mibs :` dans le fichier `/etc/snmp/snmp.conf`.

La dernière opération permet que les commandes Net-SNMP utilisent les MIB téléchargées et affichent les noms des objets plutôt que leurs OID (p.ex., sysName au lieu de 1.3.6.1.2.1.1.5).

Le fichier de configuration de `snmpd` (l'agent qui répond aux requêtes du NMS) est `/etc/snmp/snmpd.conf`.

I 1.6 Sur la machine agent, supprimez puis recréez ce fichier avec le contenu minimal ci-dessous :

```
agentAddress udp:161
rocommunity public
rwcommunity private
```

Exercice 2 — Surveillance des processus

Il est possible avec les outils Net-SNMP de déclencher l'envoi de notifications suite à la modification des objets de la MIB. Pour pouvoir observer les données relatives aux processus, aux disques durs, à la charge du système, . . ., les outils Net-SNMP utilisent la MIB UCD (nommée ainsi car initialement conçue à l'Université de Californie, Davis). Cette une MIB privée dont l'OID est 1.3.6.1.4.1.2021.

Nous allons voir dans cet exercice comment, grâce à cette MIB, observer un service en exécution et envoyer une notification en cas d'arrêt du service. Nous prendrons le service `sshd` comme exemple.

I 2.1 Définissez, dans le fichier de configuration de l'agent, une communauté pour l'envoi de notifications ainsi que la destination de ces notifications, comme nous l'avions fait dans le TP précédent :

```
trapcommunity <communaute-de-notification>
trap2sink <IP-du-NMS>
```

Pour la communauté, vous choisirez un nom quelconque.

I 2.2 Ajoutez les quatre lignes ci-dessous dans le fichier de configuration de l'agent :

```
createUser      user
iquerySecName   user
agentSecName    user
rouser          user
```

(Ces lignes créent un utilisateur SNMPv3 appelé `user` ayant le droit d'interroger l'agent. Ces définitions sont nécessaires même si dans ce TP nous utilisons uniquement SNMPv2c.)

Le fichier `/lib/systemd/system/snmpd.service` de l'agent détermine la façon dont est lancé et arrêté le service `snmpd`. Nous allons le modifier car, dans l'état actuel, il ne permet pas d'activer la surveillance de la MIB.

I 2.3 Remplacez dans ce fichier la ligne `Environment="MIBS="` par `Environment="MIBS=ALL"`. (Ignorez ce point si la ligne n'est pas présente.)
I 2.4 Dans ce même fichier, enlevez, dans la ligne `ExecStart=...`, le paramètre `-I -smux,mteTrigger,mteTriggerConf`.
I 2.5 Exécutez la commande ci-dessous pour que les modifications soient prises en compte :

```
# systemctl daemon-reload
```

On peut maintenant configurer la surveillance du processus `sshd`.

I 2.6 Ajoutez la ligne ci-dessous dans le fichier de configuration de l'agent :

```
proc sshd
```

Cette ligne signifie qu'il faut qu'un processus nommé `sshd` soit toujours présent sur la machine.

I 2.7 Démarrez le service `sshd` s'il n'est pas lancé.

I 2.8 Récupérez, avec `snmpwalk`, la branche 1.3.6.1.4.1.2021.2. N'utilisez pas l'option `-On` qui affiche les OID sous forme numérique afin de voir les noms des objets récupérés. La branche récupérée devrait contenir des objets créés suite à l'utilisation de la directive `proc`. On voit normalement le nom du processus observé (`prNames`) ainsi qu'un *drapeau* d'erreur (`prErrorFlag`) qui vaut 0 (⇔ pas d'erreur) puisque `sshd` est en exécution.

La supervision d'un service se fait par l'ajout d'un *moniteur*.

I 2.9 Ajoutez les lignes ci-dessous dans le fichier de configuration de l'agent :

```
notificationEvent trapService 1.2.3.1.4.1.1000.10.1 -o prNames -o prErrMessage
monitor -r 10 -e trapService "erreur service" prErrorFlag != 0
```

La seconde ligne peut s'interpréter de cette façon : toutes les 10 secondes (`-r 10`), tester si l'objet `prErrorFlag` est différent de 0 et, si c'est le cas, déclencher l'événement `trapService`. La chaîne de caractère `"erreur service"` est un nom associé au moniteur. L'événement `trapService` est défini à la première ligne. Il consiste à envoyer une notification SNMP ayant l'OID 1.2.3.1.4.1.1000.10.1 et contenant le nom du processus arrêté (`-o prNames`) et le message d'erreur (`-o prErrMessage`), c'est-à-dire les objets renvoyés par la commande `snmpwalk` utilisée précédemment. (*Remarque.* La notification 1.2.3.1.4.1.1000.10.1 n'existe pas dans la MIB. Il faudrait en théorie la déclarer dans la MIB avec le langage de description SMI vu en cours.)

I 2.10 Tout en capturant les trames, arrêtez le serveur `sshd` sur l'agent. On devrait normalement capturer la notification.

La directive `proc` permet aussi de préciser le nombre de processus identiques (ayant le même nom) que l'on souhaite avoir en exécution. Comme le nombre de processus `sshd` dépend du nombre de connexions SSH ouvertes sur la machine (c'est exactement $1 + 2\times$ le nombre de connexions) on peut utiliser ce mécanisme pour recevoir une notification s'il y a trop de connexions ouvertes simultanément. Pour pouvoir tester simplement, on configurera `snmpd` de sorte qu'une notification soit envoyée à la première connexion ouverte (donc avec au maximum 1 processus `sshd`).

I 2.11 Redémarrez le service `sshd`.

I 2.12 En consultant le manuel de `snmpd.conf`, trouvez comment préciser que l'on souhaite toujours avoir un et un seul processus `sshd`.

I 2.13 Vérifier qu'une notification est bien envoyée lorsque l'on ouvre connexion SSH depuis le NMS sur la machine agent (`ssh <ip-de-agent>`). En effet, à ce moment là, le nombre de processus `sshd` en exécution sur l'agent sera de 3, ce qui devrait provoquer l'envoi de la notification.

Exercice 3 — Surveillance du disque

Il est aussi possible de surveiller la taille des fichiers ou des répertoires. Cela peut être utile, par exemple, pour s'assurer que le fichier d'une base de données n'occupe pas trop de place sur le disque. On utilise alors la directive `file`.

I 3.1 Ajoutez dans le fichier de configuration de l'agent la ligne ci-dessous :

```
file <chemin-absolu-du-fichier> <taille-max-du-fichier-en-kilo-octets>
```

Choisissez le chemin d'un fichier inexistant (nous le créerons par la suite) et une taille pas trop grande.

Comme précédemment, des objets donnant des informations sur le fichier observé ont été créés dans la MIB.

I 3.2 Vérifiez, avec `snmpwalk` que la branche 1.3.6.1.4.1.2021.15 est bien présente avec les informations adéquates.

I 3.3 En vous inspirant de ce qui a été fait au point I 2.9 de l'exercice précédent et en observant le résultat de la commande `snmpwalk` utilisé au point précédent, faites en sorte qu'une notification soit envoyée quand la taille du fichier excède la taille maximale fixée. La notification devra contenir trois objets : le nom du fichier observé, sa taille actuelle et le message d'erreur. Observer le résultat de la commande `snmpwalk` pour trouver le nom de ces objets. Comme dans l'exercice précédent, utilisez un OID de notification inexistant.

I 3.4 Testez et vérifiez avec wireshark que la notification a bien été envoyée.

Exercice 4 — Traitement des notifications sur le NMS

Sous Linux, il existe le service `snmptrapd` pour traiter les notifications reçues. Son fichier de configuration est `/etc/snmp/snmptrapd.conf`. Nous allons le configurer pour pouvoir traiter les notifications reçues par le NMS.

I 4.1 Sur le NMS, modifiez le fichier de configuration de `snmptrapd` pour qu'il contienne uniquement la ligne ci-dessous :

```
authCommunity execute <communaute-de-notifications>
traphandle default /bin/traitement-notification
```

La première ligne signifie qu'à chaque notification reçue provenant de la communauté précisée, le NMS exécutera une commande. La deuxième ligne définie la commande qui sera exécutée : c'est le script `/bin/traitement-notification`. C'est donc ce script (que nous écrirons par la suite) qui va traiter les notifications reçues. Le mot `default` signifie que le programme sera appelé quelle que soit la notification reçue. On peut le remplacer par un OID si l'on souhaite appeler ce programme uniquement dans le cas de la réception d'une notification ayant un OID spécifique.

À chaque invocation d'un script de traitement de notifications, `snmptrapd` lui envoie sur son entrée standard :
— une 1ère ligne contenant le nom de l'équipement source de la notification ;
— une 2ème ligne contenant les IP et ports source et destination ;
— et, pour chaque objet contenu dans la notification, une ligne avec son OID et sa valeur.

Pour tester, on partira du script minimal ci-dessous :

```
#!/bin/bash
read nom
echo "Nom : "$nom
read ip
echo "IP et ports : "$ip
while read obj ; do
    echo "Objet : "$obj
done
```

I 4.2 Créez le script `/bin/traitement-notification` avec ce contenu.
I 4.3 Arrêtez le service `snmptrapd`.
I 4.4 Lancer `snmptrapd` dans un terminal (sans passer par systemctl) avec l'option `-f`. Le service ne vous rend pas la main. (On le lance de cette façon afin de voir plus facilement ce qu'il affiche. Sinon il faudrait regarder dans le journal.)
I 4.5 Générez l'envoi d'une notification par l'agent pour vérifier que les informations sont bien affichées par le script.
I 4.6 Arrêtez le processus `snmptrapd`. (Dans la suite, vous pourrez à nouveau utiliser `systemctl` pour lancer `snmptrapd`.)

L'objectif final est d'écrire un script de traitement des notifications permettant d'envoyer un mail à l'administrateur suite à la réception d'une notification. Ce mail devra contenir :
— la date de réception de la notification ;
— l'adresse IP et le nom de l'équipement ayant envoyé la notification ;
— et l'OID de la notification envoyée.

I 4.7 Modifiez le fichier `traitement-notification` en conséquence puis tester. Pour l'adresse mail, utiliser un service d'adresses mail jetables comme, par exemple, `https://getnada.com/`.

TP 3 — Supervision de switchs

Le TP est à faire en binôme sur 2 machines (que nous nommerons NMS et M) sur lesquelles vous restaurerez une image debian. Chaque binôme travaillera sur un switch CISCO. Les commandes de l'IOS CISCO à utiliser dans ce TP sont disponibles ici : `https://www.lipn.univ-paris13.fr/~evangelista/cours/cisco.pdf`. La machine NMS sera celle qui supervisera le switch. M l'administrera (via l'outil minicom).

Exercice 1 — Travail préparatif

On suppose que l'interface eth0 est celle connectant les machines au réseau de la salle de TP. Adaptez les instructions si ce n'est pas le cas.

I 1.1 Sur NMS : installez les paquets `snmp` et `wireshark`.
I 1.2 Sur M : installez les paquets `snmp`, `wireshark` et `minicom`.
I 1.3 Connectez les interfaces eth0 des deux machines au switch.
I 1.4 Suivez les instructions de la section 1 de la documentation cisco.

Dans la suite du TP, à chaque fois qu'il sera demandé de configurer le switch (comme, par exemple, au point suivant), cela devra être fait dans le terminal minicom.

I 1.5 Suivez les instructions de la section 3.8.1 de la documentation cisco pour donner au switch une IP (sur le VLAN 1) dans le réseau $10.24.G.0/24$ (avec G = numéro de groupe).
I 1.6 Donnez aux interfaces eth0 des 2 machines des IP dans ce même réseau.
I 1.7 Vérifiez que les pings passent bien entre les machines et le switch.

Exercice 2 — Activation de l'agent SNMP

Nous allons maintenant activer l'agent SNMP du switch pour que nous puissons ensuite le surveiller par SNMP. Deux communautés SNMP pourront interroger l'agent : l'une en lecture seule et l'autre en lecture/écriture. Nous allons utiliser le mécanisme des ACL pour restreindre les communautés SNMP de la manière suivante :
— La communauté en lecture/écriture se réduit au seul NMS.
— La communauté en lecture seule contient toutes les machines du réseau $10.24.G.0/24$ sauf le NMS.

I 2.1 Suivez les instructions de la section 3.9 de la documentation cisco pour créer deux ACL répondant aux contraintes énoncées, une pour chaque communauté.
I 2.2 Suivez les instructions de la section 3.10.1 de la documentation cisco pour créer deux communautés avec des noms de votre choix en les associant aux ACL créées.
I 2.3 Effectuez, depuis M et le NMS, plusieurs requêtes SNMP pour tester votre configuration. Vous choisirez des tests permettant de montrer que les contraintes énoncées sont bien respectées.

Q 2.1 Justifiez le choix des tests effectués.

Exercice 3 — Activation de syslog

Syslog est un protocole utilisé pour centraliser les journaux. C'est un outil important dans le domaine de la supervision car il facilite le travail de l'administrateur en regroupant tous les journaux des équipements supervisés sur une même machine. Il est de plus disponible sur de nombreux équipements et sur la plupart des systèmes d'exploitation. Son principe est simple : à chaque fois qu'un équipement écrit des informations dans son journal, il envoie également un message syslog contenant cette information à un superviseur (le NMS dans notre cas). Celui-ci regroupe dans son propre journal les informations envoyées par les équipements supervisés. Un niveau de gravité allant de 0 (critique) à 7 (information de débuggage) est associé à chaque événement. Sous Debian, c'est le service rsyslog qui traite les messages syslog reçus et le fichier de journal est `/var/log/syslog`.

I 3.1 Sur le switch : suivez les instructions de la section 3.11 de la documentation cisco pour activer l'envoi des logs vers le NMS avec un niveau maximal de notification fixé à 5.
I 3.2 Sur le NMS : trouvez, puis décommentez, dans le fichier de configuration de `rsyslog` (`/etc/rsyslog.conf`),

les deux lignes qui activent la réception de messages syslog sur le port UDP 514.

I 3.3 Débranchez puis rebranchez le câble ethernet entre M et le switch.

I 3.4 Vérifiez que le journal (`/var/log/syslog`) du NMS contient bien deux nouvelles lignes.

Exercice 4 — Activation des notifications

On veut maintenant activer sur le switch l'envoi de notifications SNMP au NMS. Le switch est capable d'envoyer des notifications SNMPv2c standard (celles définies dans la branche 1.3.6.1.6.3.1.1.5) ainsi que de nombreuses notifications propres aux équipements CISCO.

I 4.1 Sur le switch, suivez les instructions de la section 3.10.2 de la documentation cisco pour programmer l'envoi de notifications SNMP au NMS dans les situations suivantes :
- suite au branchement ou au débranchement d'un câble ;
- et suite à la création d'un VLAN.

Les notifications seront envoyées au format SNMPv2c avec une communauté de votre choix.

I 4.2 Testez avec wireshark que des notifications sont bien reçues dans les trois cas.

Q 4.1 Pour chaque notification capturée, dites si c'est une notification SNMPv2c standard ou si c'est une notification propre aux équipements CISCO. Justifiez.

Exercice 5 — Droits d'accès à la MIB

Actuellement, seul un contrôle sur l'adresse IP et la communauté est fait pour savoir si une machine peut lire ou modifier la MIB. On peut définir des droits de manière plus fine grâce aux *vues* afin que les machines puissent avoir accès à certaines parties de la MIB uniquement. Nous allons les utiliser pour définir les droits d'accès de M à la MIB de la manière suivante :

— accès à la branche MIB-2 en lecture seule
— accès à la branche system en lecture/écriture
— aucun accès au reste de la MIB

I 5.1 Créez une nouvelle ACL contenant uniquement M ainsi que les vues et communautés répondant aux contraintes énoncées. Une communauté ne pouvant être associée qu'à une seule vue, il est nécessaire de définir deux communautés pour M. La première sera utilisée pour accéder à la branche MIB-2 en lecture seule ; la deuxième pour accéder à la branche system en lecture/écriture. Les instructions pour créer les vues sont décrites dans la section 3.10.3 de la documentation cisco.

I 5.2 Proposez et lancez une série de tests (sous forme de requêtes SNMP) sur M permettant de tester que votre configuration répond aux contraintes énoncées. Dans vos tests, vous pourrez utiliser l'objet 1.3.6.1.2.1.11.30 qui est modifiable, dans la branche MIB-2 mais pas dans la branche system.

Q 5.1 Justifiez les choix des tests effectués.

TP 4 — Syslog sous linux avec le service rsyslog

Ce TP est à faire sous marionnet. On rappelle que le fichier du service rsyslog est `/etc/rsyslog.conf`. Pensez à redémarrer le service après chaque modification de ce fichier. Pour redémarrer, arrêter ou vérifier le statut d'un service :

```
# /etc/init.d/<nom-du-service> restart
# /etc/init.d/<nom-du-service> stop
# /etc/init.d/<nom-du-service> status
```

(La commande `systemctl` n'est pas disponible sur les machines marionnet qui sont des debian utilisant le système d'initialisation SysV init (plutôt que systemd qui est maintenant le système utilisé par la plupart des distributions).)

Vous utiliserez la commande logger (voir cours) pour vérifier les configurations de syslog sur les différentes machines aux exercices 2 à 5.

Rappel : `tail -1 <fichier>` permet de voir la dernière ligne du fichier (pour voir le dernier message syslog enregistré).

Exercice 1 — Création du réseau

Créer un réseau composé de quatre machines reliées à un switch :

— syslog — un serveur rsyslog qui stockera les messages de journal envoyés par les deux machines ci-dessous ;
— client1 et client2 — deux PC dont on souhaite collecter les journaux ;
— snmp — un NMS qui réceptionnera des notifications SNMP envoyées par le serveur syslog.

Choisir l'image par défaut lors de la création des machines et donner aux machines des IP dans le réseau $10.24.N.0/24$, N étant votre numéro de poste dans la salle de TP.

Exercice 2 — Première analyse du fichier de configuration de rsyslog

I 2.1 Démarrer le service rsyslog sur la machine syslog.
I 2.2 Ouvrir le fichier de configuration de rsyslog (`/etc/rsyslog.conf`) sur la machine syslog avec la commande `less`. (Pour quitter `less` par la suite : touche q.)
I 2.3 Rechercher le mot RULES en tapant `/RULES` puis entrée.

Sous la ligne de commentaire contenant ce mot sont définies les règles indiquant comment sont traités les messages syslog reçus par le serveur. On s'intéressera uniquement aux 11 premières règles (toutes les règles jusqu'à la règle `mail.err` ... incluse). En analysant ces règles indiquer dans quel(s) fichier(s) seront écrits les messages syslog suivant :

— les messages de la catégorie daemon
— les messages de la catégorie auth
— les messages de la catégorie mail et de gravité alert
— les messages de la catégorie mail et de gravité warning

Donner, pour chaque cas, le (ou les) ligne(s) qui s'appliquent.

Exercice 3 — Configuration de rsyslog sur les deux clients

Nous allons maintenant configurer rsyslog sur client1 et client2.

— Sur les deux clients : supprimer ou commenter les règles déjà présentes dans le fichier de configuration (toutes les lignes après le commentaire `### RULES ###`).
— client1 propose différents services (http, ftp, ...). On souhaite donc configurer le service rsyslog sur client1 selon les spécifications suivantes :
  1. stocker les messages de la catégorie `daemon` dans le fichier `/var/log/daemon.log` ;
  2. stocker tous les messages sauf ceux de la catégorie `daemon` dans le fichier `/var/log/messages.log` ;
  3. et rediriger les messages de la catégorie `daemon` ayant un niveau `error` (ou plus grave) vers la machine syslog en utilisant UDP.
— client2 propose un service ssh. On souhaite donc configurer le service rsyslog sur client2 selon les spécifications suivantes :
  1. rediriger les messages de la catégorie `auth` ayant un niveau `critical` (ou plus grave) vers la machine syslog en utilisant UDP ;
  2. stocker les messages de la catégorie `auth` ayant un niveau égal à `error` dans le fichier `/var/log/auth-error.log` ;
  3. stocker les messages de la catégorie `auth` ayant un niveau de gravité autres que ceux décrits dans les deux premières clauses dans le fichier `/var/log/auth-pas-grave.log`.

Remarque : Pour l'instant, on ne configurera pas la machine syslog pour qu'elle traite les messages envoyés par les deux clients. On utilisera donc tcpdump (capteur de paquets en mode texte) ou wireshark sur le serveur pour vérifier que les messages syslog sont bien reçus.

Exercice 4 — Configuration du serveur rsyslog

Sur la machine syslog, configurer le serveur rsyslog selon les spécifications suivantes :

1. Le serveur accepte les messages envoyés avec UDP. Il faut pour cela trouver puis décommenter deux lignes en début de fichier qui activent la réception de messages UDP sur le port 514.
2. Tout message (reçu ou généré par la machine syslog) est stocké dans le fichier `/var/log/messages.log`.
3. Les messages (reçus ou générés par la machine syslog) des catégories `auth` et `daemon` seront stockés dans les fichiers `/var/log/auth.log` et `/var/log/daemon.log` respectivement.

Exercice 5 — Envoi de notifications SNMP

Faire en sorte, à l'aide d'un tube, que la machine syslog envoie une notification SNMP à la machine snmp pour tout message (reçu ou généré par la machine syslog) de la catégorie `auth` ou `daemon` et de niveau `emerg`.

Vous devrez écrire un script bash (par exemple, `/usr/local/bin/rsyslog-snmp`) qui lira les données envoyées sur le tube et qui, pour chaque ligne lue, enverra une notification SNMP à l'aide de la commande `snmptrap`. Le script ne devra jamais terminer : tout le code sera placé dans une boucle infinie afin que le script lise en continu les données envoyées dans le tube par rsyslog.

La notification envoyée sera au format SNMPv2c. Elle utilisera une communauté de votre choix. Son sysUpTime (pour rappel, le premier objet se trouvant dans toute notification SNMPv2c) sera fixé à 0 et son snmpTrapOid (OID de la notification envoyée) vaudra 1.3.6.1.4.1.10000.1 (on imagine, pour simplifier, que la branche d'entreprise 1.3.6.1.4.1.10000 contient des OID permettant l'envoi de messages syslog dans des notifications). La notification envoyée contiendra également le contenu du message syslog dans l'objet 1.3.6.1.4.1.10000.2.

La syntaxe de la commandes `snmptrap` qui permet d'envoyer une notification est la suivante :

```
# snmptrap -v2c -c<communauté> <ip-destination> <valeur-de-sysUpTime> <oid-de-la-notification>
                               <oid1> <type1> <valeur1> <oid2> <type2> <valeur2> ...
```

Les paramètres optionnels de la deuxième ligne permettent d'insérer d'autres objets dans la notification en suivant la même syntaxe que `snmpset`.

En capturant les trames sur la machine snmp, vérifiez qu'une notification est bien reçue lorsqu'un message de priorité `auth.emerg` est généré sur client2.

TP 5 — Introduction à Nagios

Le TP est à réaliser en binôme sur deux PC avec une image debian 11 : *nagios*, la machine sur laquelle nous installerons l'outil de supervision nagios ; et *agent*, la machine qui sera supervisée par la machine nagios.

Compte-rendu En fin de séance, vous enverrez en guise de compte-rendu :
— le PDF contenant les réponses et copies d'écran demandées dans le sujet ;
— `TP5-NOM1-NOM2-nagios.tgz` : une archive contenant les fichiers de la machine nagios ;
— et `TP5-NOM1-NOM2-agent.tgz` : une archive contenant les fichiers de la machine agent.
Sur la machine nagios, ces deux commandes archivent les fichiers demandés et vérifient le contenu de l'archive :

```
$ tar chzf TP5-NOM1-NOM2-nagios.tgz /etc/nagios4 /usr/lib/nagios/plugins
$ tar tzf TP5-NOM1-NOM2-nagios.tgz
```

Ces mêmes commandes doivent aussi être exécutées sur la machine agent (en changeant bien sûr `nagios` par `agent`).

Exercice 1 — Travail préparatif

I 1.1 Sur la machine nagios, installez les paquets `wireshark`, `nagios4`, `nagios-nrpe-plugin` et `postfix` (serveur d'envoi de mails). À l'installation de postfix, séléctionnez *Site Internet* pour le mode d'envoi des mails.
I 1.2 Sur la machine agent, installez les paquets `wireshark`, `proftpd-basic` et `nagios-nrpe-server`.

Exercice 2 — Interface web de Nagios

L'interface web de nagios repose sur l'exécution de scripts CGI (Common Gateway Interface). Ils ne sont pas nécessairement autorisés par le serveur web apache. Il nous faut pour cela activer le module apache `cgi`.

I 2.1 Sur la machine nagios, activez le module `cgi` :

```
# a2enmod cgi
```

I 2.2 Sur la machine nagios, redémarrez le serveur apache.

L'interface web de nagios est accessible depuis l'URL `http://<ip-de-nagios>/nagios4`.

Q 2.1 Quelles sont les machines actuellement supervisées par le service nagios (voir la page *Hosts*) ?
Q 2.2 Quels sont les informations collectées et affichées sur ces machines (voir la page *Services*) ?

Exercice 3 — Supervision de la machine agent

Les fichiers de configuration de nagios se trouvent dans `/etc/nagios4`. Le fichier principal est `nagios.cfg`. Pour qu'un autre fichier de configuration soit pris en compte il faut, dans ce fichier, ajouter une ligne de la forme suivante : `cfg_file=/etc/nagios4/<fichier-a-ajouter>`

Nous allons maintenant configurer la machine nagios pour qu'elle supervise le service FTP de la machine agent, c'est-à-dire, pour qu'elle teste périodiquement si ce service est bien actif. La première étape est de déclarer un hôte (`host`).

Les fichiers décrivant les éléments supervisés par nagios sont dans le répertoire `/etc/nagios4/objects`.

I 3.1 Créez un fichier `/etc/nagios4/objects/agent.cfg` avec le contenu suivant :

```
define host {
   use          linux-server
   host_name    <nom-de-agent>
   address      <ip-de-agent>
}
```

C'est dans ce fichier que nous déclarons une nouvelle machine à superviser (avec la déclaration **host**) et que nous déclarerons par la suite les opérations de supervision que nous souhaitons effectuer sur cette machine. La ligne **use** indique que la machine hérite de certaines caractéristiques liées au type de machines `linux-server`. Le nom choisi sera celui qui apparaîtra dans l'interface web.

I 3.2 Ajoutez à la fin du fichier `nagios.cfg` une référence vers ce nouveau fichier (une ligne `cfg_file=...`).

Q 3.1 Donnez une copie d'écran de la page *Hosts*.

Actuellement l'agent est déclaré, mais nagios n'effectue aucune opération de supervision sur celui-ci (à part vérifier qu'il répond aux pings). Nous allons faire en sorte que nagios teste périodiquement la disponibilité de son service FTP.

I 3.3 Dans le fichier `agent.cfg`, ajoutez ces lignes :

```
define service {
    use                 generic-service
    host_name           <nom-de-agent>
    service_description <description-du-service-supervisé>
    check_command       check_ftp
    check_interval      1m
}
```

La ligne **check_command** indique la commande utilisée pour vérifier la disponibilité du serveur FTP. La commande `check_ftp` est une commande incluse nagios. Nous verrons dans les exercices suivants comment définir de nouvelles commandes. La ligne **check_interval** indique à quelle fréquence nagios fera le test (1 minute ici).

I 3.4 Observez les changements sur l'interface web lorsque l'on arrête ou démarre le service proftpd sur l'agent.

Q 3.2 Donnez deux copies d'écran de la page *Services* selon que proftpd est actif ou inactif.
Q 3.3 En analysant le trafic entre les deux machines, déterminez la méthode employée par nagios pour tester la disponibilité du service FTP.

Exercice 4 — Supervision du routeur de la salle de TP

En suivant les instructions de l'exercice précédent, configurez la machine nagios afin qu'elle teste toutes les minutes la disponibilité d'un serveur HTTP sur le routeur de la salle de TP. Vous créerez pour cela un nouveau fichier `routeur.cfg` dans le répertoire `/etc/nagios4/objects`, comme nous l'avons fait pour l'agent.

Q 4.1 Donnez des copies d'écran des pages *Hosts* et *Services*.

Exercice 5 — Mise en place de notifications

On souhaite maintenant que l'administrateur (vous) soit prévenu par mail en cas de changement d'état du service FTP sur la machine agent. Nous allons pour cela définir un contact et un groupe de contacts contenant celui-ci.

I 5.1 Créez le fichier `/etc/nagios4/objects/admin.cfg` avec le contenu suivant :

```
define contact {
    contact_name                    <nom-du-contact>
    email                           <adresse-mail-du-contact>
    service_notifications_enabled   <???>
    service_notification_commands   notify-service-by-email
    service_notification_period     <???>
    service_notification_options    <???>
    host_notification_commands      notify-host-by-email
}
define contactgroup {
    contactgroup_name   <nom-du-groupe>
    alias               <alias-du-groupe>
    members             <nom-du-contact>
}
```

Cherchez sur Internet la signification des paramètres ayant la valeur `<???>`. Choisissez des valeurs afin que le mail soit envoyé à tout moment si un service passe de l'état actif à inactif ou inversement (et dans ces deux cas seulement). Pour l'adresse mail, utilisez un service d'adresses jetables (p.ex., `https://getnada.com/`).

I 5.2 Ajoutez dans le fichier de configuration principal de nagios un lien vers ce nouveau fichier.

I 5.3 Ajoutez dans la déclaration du service FTP (à l'intérieur du bloc **define service** { ... }) les lignes suivantes :

```
    notifications_enabled       1
    first_notification_delay    0
    notification_interval       0
    contact_groups              <nom-du-groupe>
```

La première ligne active l'envoi de mails lors d'un changement d'état du service. La deuxième indique le temps d'attente (en secondes) avant l'envoi du mail lorsqu'un changement d'état est observé. La troisième indique le nombre de mails de rappels envoyés après le premier mail.

I 5.4 Arrêtez le service proftpd sur l'agent pour tester si le mail est envoyé. Retestez après le redémarrage du service.

Q 5.1 Donnez la signification des différents paramètres pour lesquels la valeur était `<???>`.
Q 5.2 Donnez des copies d'écran des deux mails reçus.

Exercice 6 — Développement d'un plugin

Les tâches de supervision exécutées par le service nagios sont effectuées grâce à des fichiers exécutables appelés plugins et se trouvant dans le répertoire `/usr/lib/nagios/plugins`. À l'installation de nagios, ce répertoire contient de nombreux plugins permettant de réaliser des tests usuels. La commande `check_ftp` utilisée précédemment pour vérifier la disponibilité du service FTP en est un exemple. Quand l'administrateur ne trouve pas dans ce répertoire de script répondant à ses besoins, il doit en développer de nouveaux.

Un plugin nagios doit avoir un code de retour parmi les trois valeurs suivantes : 0 (OK), 1 (avertissement ou warning) ou 2 (critique). Ces codes correspondent aux couleurs affichées dans l'interface web (0 = vert, 1 = orange et 2 = rouge).

L'objectif de cet exercice est d'écrire (en bash) un plugin `check_jitter` qui testera la gigue d'une liaison (jitter en anglais). Le script prend trois arguments : l'IP (ou le nom) de l'hôte à contacter, et deux seuils de gigue (en millisecondes) : un seuil d'avertissement et un seuil critique. Il aura un code de retour de :

— 0 si la gigue est inférieure ou égale au seuil d'avertissement ;
— 1 si la gigue est strictement supérieure au seuil d'avertissement mais inférieure ou égale au seuil critique ;
— ou 2 dans tous les autres cas (en particulier si l'hôte n'est pas accessible).

On utilisera la commande ping pour mesurer la gigue. En effet, ping fournit directement la gigue qu'il nomme *mdev*. On arrondira la gigue fournie par ping à l'entier inférieur et, pour simplifier, on considérera que les seuils fournis en arguments sont des entiers.

Par exemple, dans le résultat de l'exécution de ping ci-dessous, le résultat fourni par ping qui nous intéresse est 4 (qui apparaît en bleu dans la dernière ligne). C'est cette valeur qui devra être comparée aux deux seuils passés en argument.

```
PING 9.9.9.9 (9.9.9.9) 56(84) bytes of data.
64 bytes from 9.9.9.9: icmp_seq=1 ttl=52 time=11.6 ms
64 bytes from 9.9.9.9: icmp_seq=2 ttl=52 time=10.7 ms
64 bytes from 9.9.9.9: icmp_seq=3 ttl=52 time=17.1 ms
64 bytes from 9.9.9.9: icmp_seq=4 ttl=52 time=22.6 ms
64 bytes from 9.9.9.9: icmp_seq=5 ttl=52 time=13.1 ms
64 bytes from 9.9.9.9: icmp_seq=6 ttl=52 time=15.5 ms
64 bytes from 9.9.9.9: icmp_seq=7 ttl=52 time=21.4 ms
64 bytes from 9.9.9.9: icmp_seq=8 ttl=52 time=25.1 ms
64 bytes from 9.9.9.9: icmp_seq=9 ttl=52 time=12.6 ms
64 bytes from 9.9.9.9: icmp_seq=10 ttl=52 time=16.1 ms

--- 9.9.9.9 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 10.693/16.577/25.107/4.712 ms
```

Voici le squelette du script :

```
#!/bin/bash

# on range les arguments du script dans des variables
ip=__________
seuilWarning=__________
seuilCritique=__________

# on lance 10 demandes d'écho vers l'IP passée en argument et on sauvegarde le résultat dans /tmp/resultat_ping
__________ > /tmp/resultat_ping
if __________
then
    echo "Erreur - hôte inaccessible"
    __________ 2
else
    gigue=__________  # on extrait la gigue du résultat fourni par ping (4 dans notre exemple)
    if __________
    then
        echo "OK - gigue = __________"
        __________ 0
    elif __________
    then
        echo "Warning - gigue = __________"
        __________ 1
    else
        echo "Erreur - gigue = __________"
        __________ 2
    fi
fi
```

I 6.1 Complétez les trous dans le script `check_jitter`.

Le fichier `/etc/nagios4/objects/commands.cfg` contient les définitions des commandes de supervision pouvant être lancées par le service nagios.

I 6.2 Ajoutez à la fin du fichier `commands.cfg` la définition de la commande nagios `check_jitter` :

```
define command {
    command_name check_jitter
    command_line /usr/lib/nagios/plugins/check_jitter $HOSTADDRESS$ $ARG1$ $ARG2$
}
```

La ligne **command_line** définit la commande qui sera exécutée par nagios. `$ARG1$` et `$ARG2$` seront remplacés par la valeur du premier et deuxième argument lorsque nous définirons (instruction suivante) le service utilisant cette commande. `$HOSTADDRESS$` sera remplacé par son IP.

I 6.3 Modifiez le fichier `agent.cfg` pour indiquer que nous allons maintenant utiliser la commande définie dans le point précédent sur la machine agent pour surveiller la gigue entre la machine nagios et la machine agent. Il faut pour cela définir un nouveau service comme nous l'avons fait dans l'Exercice 3. La commande nagios que nous avons définie au point précédent prend deux arguments (`$ARG1$` et `$ARG2$`). Pour indiquer les valeurs de ces arguments, il faut les faire précéder par un point d'exclamation, comme ceci :

```
    check_command       check_jitter!2!10
```

On utilise ici un seuil d'avertissement de 2 ms et un seuil critique de 10 ms.

I 6.4 Pour tester, utilisez la commande `tc` (traffic controller) qui permet d'introduire un délai aléatoire sur une interface. Par exemple, pour introduire un retard aléatoire de 10 ± 5 millisecondes (en root) :

```
# tc qdisc add dev eth0 root netem delay 10ms 5ms
```

Une fois cette commande exécutée, il faut changer `add` par `change` si l'on souhaite modifier le retard introduit, ou par `del` si l'on souhaite le supprimer.

Q 6.1 Donner trois copies d'écran de la page *Services*, une pour chacun des trois états possible du service.

Exercice 7 — Exécution de commandes à distance

NRPE (Nagios Remote Plugin Executor) est une extension permettant à nagios d'exécuter des commandes à distance sur les machines supervisées (un peu comme si le service nagios pouvait se connecter en SSH sur la machine supervisée). Sur la machine supervisée, le service permettant d'exécuter des commandes est nagios-nrpe-server. Ainsi, le service nagios s'exécutant sur la machine nagios demande au service nagios-nrpe-server s'éxécutant sur la machine agent d'exécuter une commande particulière. Si la machine nagios est autorisée à le faire, nagios-nrpe-server exécute cette commande et retourne le résultat à nagios. C'est le plugin `check_nrpe` qui permet à nagios d'exécuter à distance des commandes sur d'autres machines.

Nous allons utiliser cette extension pour que la machine nagios puisse périodiquement tester si le service cron (permettant de programmer l'exécution de tâches) est actif sur la machine agent. Comme le service cron n'est pas un service réseau, cette opération n'est pas possible, avec nagios, sans NRPE.

Tous les scripts pouvant être exécutés à distance par la machine nagios doivent, comme les plugins, être placées dans le répertoire `/usr/lib/nagios/plugins`.

I 7.1 Sur la machine agent :

(a) Écrivez un script `check_cron` qui teste si le service cron est actif. Le script devra afficher un message indiquant l'état du service. Son code de retour sera 0 si le service est actif et 2 si le service est inactif. (Il n'y a pas d'état "avertissement" dans ce cas.)

(b) Autorisez la machine nagios à exécuter des commandes à distance en modifiant le paramètre `allowed_hosts` du fichier de configuration `/etc/nagios/nrpe.cfg`.

(c) Dans ce même fichier de configuration, ajoutez une ligne définissant la commande distante `check_cron` :

```
command[check_cron]=/usr/lib/nagios/plugins/check_cron
```

(d) Redémarrez le service `nagios-nrpe-server`.

I 7.2 Sur la machine nagios :

(a) Vérifiez qu'il est possible d'appeler la commande `check_cron` sur la machine agent :

```
$ /usr/lib/nagios/plugins/check_nrpe -H <ip-de-agent> -c check_cron
```

(b) Dans le fichier `commands.cfg`, définissez une nouvelle commande qui utilise le plugin `check_nrpe` pour appeler la commande `check_cron` sur une machine distante, comme nous l'avons fait au point précédent.

(c) Dans le fichier `agent.cfg`, définissez un nouveau service utilisant la commande définie au point précédent pour tester la disponibilité du service cron sur la machine agent.

I 7.3 Observez sur l'interface web les changements lorsque le service cron est actif ou inactif sur la machine agent.

Q 7.1 Donner deux copies d'écran de la page *Services*, l'une avec le service cron actif l'autre avec le service inactif.