TP8 : Installation et configuration de Rsyslog et NSCA

Nagios®

Table des matières

TP8 : Installation et configuration de Rsyslog et NSCA

8.1 – Principe

8.2 – Architecture

8.2.1 – Architecture préconisée
8.2.2 – Architecture mise en œuvre pour le TP

8.3 – Ntpdate (un client Ntp)

8.3.1 – Installation de Ntpdate
8.3.2 – Configuration de Ntpdate

8.4 – Rsyslog et filtre simple

8.4.1 – Installation du serveur Rsyslog
8.4.2 – Configuration du serveur Rsyslog
8.4.3 – Installation du client Rsyslog
8.4.4 – Configuration du client Rsyslog
8.4.4 – Test

8.5 – Rsyslog et filtre faisant appel à un script.

8.5.1 – Configuration du serveur Rsyslog
8.5.2 – Script
8.5.2 – Principe

8.6 – Nsca

8.6.1 – Installation de Nsca côté serveur
8.6.2 – Installation de Nsca côté machine à superviser
8.6.3 – Fonctionnement côté machine à superviser

8.7 – Définition du service Nagios

Objectif : Ce document va nous permettre de coupler Nagios avec Rsyslog pour superviser les logs.

 **INFO :** C'est juste de l'information mais qui peut être nécessaire à la compréhension du TP

 **IMPORTANT** : Information importante à prendre en compte

 **TP à Faire :** Travaux à réaliser

TP8 : Installation et configuration de Rsyslog et NSCA

L'entreprise dans laquelle vous êtes a besoin d'avoir l'horloge de ses serveurs synchronisée avec un serveur de temps (NTP). La désynchronisation de son horloge aura un impact important dans le fonctionnement de l'entreprise.

Nous allons utiliser les logs pour vérifier si le client NTP met bien à jour la date sur le serveur lui-même, si ce n'est pas le cas, envoyer une alerte critique à Nagios. .

Ce principe pourrait être utilisé dans une multitude de cas.

8.1 – Principe

Nous allons installer et configurer le client NTP sur le serveur distant puis centraliser les logs de ce dernier sur un serveur de logs (Dans notre cas, le serveur de logs sera le même que le serveur Nagios).

Le client NTP va se synchroniser toutes les heures à un serveur de temps. Il va marquer sa synchronisation dans les logs. Si dans les logs on ne voit pas le client se synchroniser, il faut alerter Nagios pour remonter une alerte d'un problème de synchronisation.

Logiciels :

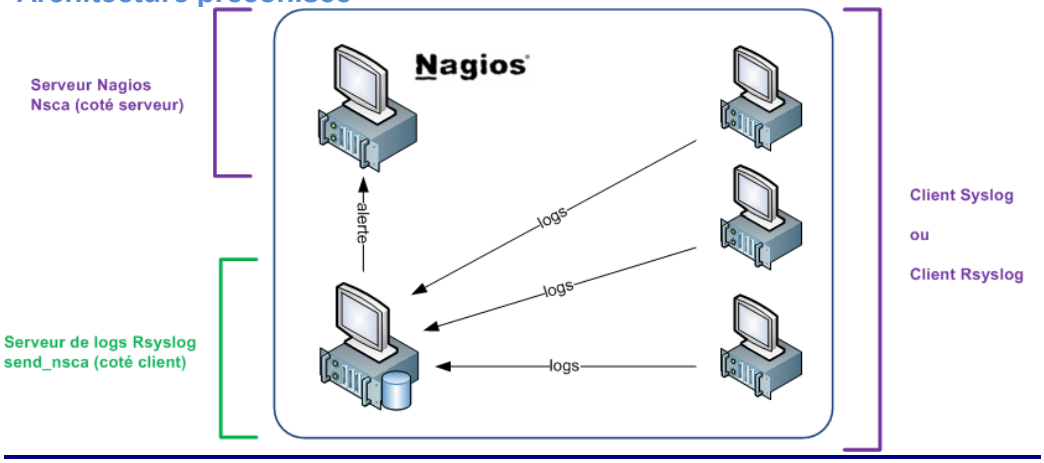
Rsyslog : Serveur de logs

Ntpdate : Client Ntp

Nsca : Agent passif qui retourne un contrôle à Nagios

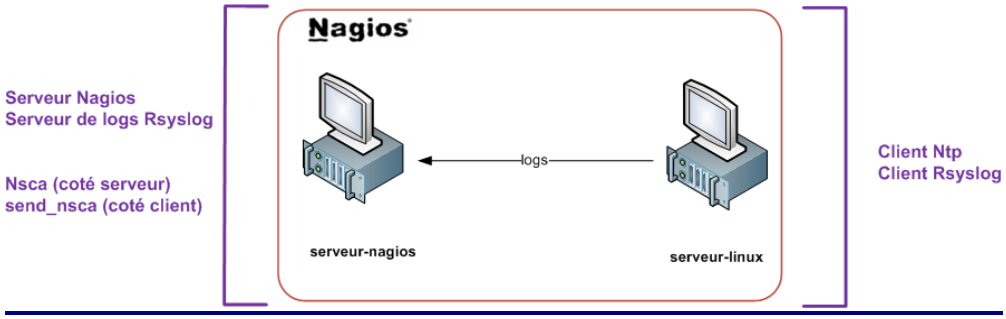
8.2 – Architecture

8.2.1 – Architecture préconisée



Les différents clients envoient leurs logs sur un serveur de logs qui filtre ces derniers. Le serveur de logs envoie l'alerte en utilisant le client Nsca (send_nsca) au serveur Nsca qui transmet à Nagios.

8.2.2 – Architecture mise en œuvre pour le TP



Pour simplifier notre configuration pour les travaux pratiques, nous avons notre serveur linux qui envoie les logs vers le serveur de logs (le même serveur que Nagios) qui contient le client Nsca (send_nsca) ainsi que le serveur Nsca.

8.3 – Ntpdate (un client Ntp : serveur-linux)

8.3.1 – Installation de Ntpdate

Pour rappel, NTP, ou Network Time Protocol est une méthode pour maintenir l'heure d'un serveur à jour. Ici nous allons utiliser le client **ntpdate** pour ce faire.

- Installer ntpdate

```
# apt-get install ntpdate
```

Configuration d'un cron pour synchroniser l'heure. La mise en place du cron va permettre au serveur NTP de se connecter avec un serveur NTP public pour synchroniser son horloge à intervalles réguliers.

8.3.2 – Configuration de Ntpdate

- Télécharger un cron que vous allez mettre dans /etc/cron.hourly, il sera donc exécuté toutes les heures.

```
# cd /etc/cron.hourly/
# wget http://howto.landure.fr/gnu-linux/debian-sarge/configuration-dun-client-ntp/ntpdate
```

 - Donner des droits d'exécution.

```
# chmod +x /etc/cron.hourly/ntpdate
```

Lors de l'exécution de la tâche, vous voyez apparaitre dans les logs (fichier de logs : /var/log/syslog) une ligne de ce genre qui permet de signaler que le serveur c'est bien mis à jour :

Fichier syslog

```
tep time server 91.121.60.158 offset -0.012709 sec
```

Désormais votre serveur se met à jour à un intervalle de temps régulier.

Nous allons maintenant installer notre serveur de logs sur lequel nous centraliserons les logs de Ntp en s'appuyant sur le logiciel Rsyslog.

8.4 – Rsyslog et filtre simple

8.4.1 – Installation du serveur Rsyslog (serveur-nagios)

 Sur Debian la gestion des logs se fait via Rsyslog qui est installé par défaut. Si celui-ci n'est pas installé, vous pouvez l'installer en faisant :

```
# apt-get install rsyslog
```

Ici nous allons configurer Rsyslog pour qu'il puisse servir de serveur de logs et permettre la centralisation des logs de différents serveurs et l'analyse de ces derniers.

8.4.2 – Configuration du serveur Rsyslog

 - Editer le fichier **/etc/rsyslog.conf**

Pour la configuration tout passe par le fichier de configuration situé dans /etc. Afin d'activer la réception des logs via le réseau il faut simplement décommenter quelques lignes : on ajoute le support udp et tcp.

Fichier rsyslog.conf

```
$ModLoad imupd
$UDPServerRun 514

$ModLoad imtcp
$InputTCPServerRun 514
```

Filtre simple :

 Pour commencer nous allons mettre en place un filtre simple pour se familiariser avec le système de filtrage de Rsyslog (Ce n'est pas le filtre qui sera utilisé pour le filtrage final). Nous ajoutons dans le fichier rsyslog.conf les directives ci-dessous :

Fichier rsyslog.conf

```
$template ntp, "/home/nagios/log/%HOSTNAME%/ntp"
:msg, contains, "step time server" -?ntp
```

Nous venons de faire un filtre sur une chaine de caractères.

 - Redémarrer rsyslog

```
# /etc/init.d/rsyslog restart
```

8.4.3 – Installation du client Rsyslog (serveur-linux)

 Sur Debian la gestion des logs se fait via Rsyslog qui est installé par défaut. Si celui-ci n'est pas installé, vous pouvez l'installer en faisant :

```
# apt-get install rsyslog
```

Ici nous allons configurer Rsyslog pour qu'il puisse envoyer ses logs vers un serveur de logs qui lui, les filtrera.

8.4.4 – Configuration du client Rsyslog

 Maintenant que le **filtre simple** est en place sur le serveur de logs, il vous suffit de définir le serveur vers lequel vous voulez envoyer les logs, dans le fichier rsyslog.conf en indiquant l'adresse IP du serveur de logs.

Fichier rsyslog.conf

```
*.* @IP-du-serveur-de-logs
```

. Permet d'indiquer tous les logs (type et priorité).
@ A mettre devant l'adresse IP.

 - Redémarrer rsyslog

```
# /etc/init.d/rsyslog restart
```

8.4.5 – Test

 Lancer sur le serveur distant (client Ntp : serveur-linux) la commande /etc/cron.hourly/ntpdate pour forcer la synchronisation de l'heure.

 **Compte rendu-17 : Expliquer le fonctionnement du filtre simple et le résultat obtenu après avoir filtré les logs provenant de l'exécution du cron sur le client Ntp (cron qui permet de forcer la synchronisation de l'heure)?**

8.5 – Rsyslog et filtre faisant appel à un script.

8.5.1 – Configuration du serveur Rsyslog (serveur-nagios)

Filtre faisant appel à un script qui renvoie une alerte à Nagios :

Nous allons maintenant créer un filtre qui exécutera un script si les logs filtrés correspondent à la chaine de caractère définie dans ce filtre :

 Nous ajoutons dans le fichier rsyslog.conf les directives ci-dessous :

```
$template nagios-ntp, "%HOSTNAME%;ntp_critical;0;NTPD OK - %msg%"
:msg, regex, "step time server" ^/usr/local/nagios/libexec/rsyslog-
nagios.sh;nagios-ntp
```

$template : Nous avons créé un gabarit qui fixe le format du message pour le script **rsyslog-nagios.sh**

Vient ensuite la directive qui fixe le filtre et ce qui est fait des messages correspondants.

Si dans le corps du message on tombe sur la chaine de caractère **"step time server"**, alors on exécute le script **rsyslog-nagios.sh** avec comme paramètre le nom de la machine de l'hôte, le nom du service, un code retour et le contenu du message.

 - Redémarrer rsyslog

```
# /etc/init.d/rsyslog restart
```

8.5.2 – Script

Le script **rsyslog-nagios.sh :**

```
#!/bin/bash

CMD="/usr/local/nagios/bin/send_nsca"

CFG="/usr/local/nagios/etc/send_nsca.cfg"

PART1=$(echo $1 | cut -d";" -f1)
PART2=$(echo $1 | cut -d";" -f2)
PART3=$(echo $1 | cut -d";" -f3)
PART4=$(echo $1 | cut -d";" -f4)

/bin/echo "$PART1","$PART2","$PART3","$PART4" | $CMD -H 127.0.0.1 -c $CFG -d ","
```

L'option –H permet de définir le serveur Nagios.
L'option –d permet de modifier le délimiteur, ici nous avons utilisé la virgule.

8.5.3 – Principe

Nous voulons centraliser les logs du client Ntpdate qui sont générés à chaque synchronisation avec le serveur NTP. Et nous pouvons voir qu'à chaque synchronisation, le client Ntpdate envoie un message de synchronisation et la ligne suivante apparait dans les logs rsyslog :

```
step time server 91.121.60.158 offset -0.012709 sec
```

Le raisonnement suivi est que si nous détectons cette chaine de caractères toutes les heures, cela veut dire que le serveur est fonctionnel parce qu'il arrive à se mettre à jour.
Si ce message n'est pas détecté au bout de 70 minutes (1heure + 10 minutes), le serveur peut avoir un problème, Il faut donc envoyer un message d'alerte à Nagios

Si vous regardez bien le script, pour envoyer les alertes sur Nagios il utilise Nsca…Dans la partie suivante nous allons voir son installation et son paramétrage.

8.6 – Nsca

Maintenant que nous avons centralisé les logs, il est temps d'interagir avec Nagios. Nous allons pour cela utiliser l'outil Nsca.

 Le principe est simple : il est constitué » d'un serveur Nsca (/etc/nagios/nsca.cfg) qui est installé sur le serveur Nagios et d'un client Nsca (/etc/nagios/send_nsca.cfg) installé sur le serveur de logs effectuant des vérifications passives (dans notre contexte TP, il sera sur le même serveur : serveur-nagios).

8.6.1 – Installation de Nsca côté serveur (serveur-nagios)

 Prérequis :

```
# apt-get install libmcrypt4 snmp
```

 - Télécharger et compiler :

```
# wget http://downloads.sourceforge.net/project/nagios/nsca-2.x/nsca-2.9.1/nsca-
2.9.1.tar.gz

# tar -xzf nsca-2.9.1.tar.gz

# cd nsca-2.9.1/

# ./configure

# make
```

 - Copier le binaire et le fichier de configuration qui nous intéresse pour la partie serveur.

```
# cp src/nsca /usr/local/nagios/bin/

# cp sample-config/nsca.cfg /usr/local/nagios/etc/
```

 Quelques explications sur le fichier de configuration (nsca.cfg) :

```
# METTRE LE MOT DE PASSE QUE VOUS DESIREZ
password=xxxxxxxx

# LA METHODE DE CRYPTAGE (FAITES VOTRE CHOIX)
```

```
decryption_method=1
```

 Pour le lancement de Nsca, on va opter pour la méthode démon comme (ci-dessous). Vous devez installer le script d'arrêt/démarrage de Nsca dans /etc/init.d :

```
#!/bin/sh

# simple debian init script for nsca
# by sean finney <seanius@debian.org>

DAEMON=/usr/local/nagios/bin/nsca
NAME=nsca
DESC="Nagios Service Check Acceptor"
CONF=/usr/local/nagios/etc/nsca.cfg
OPTS="--daemon -c $CONF"
PIDFILE="/var/run/nsca.pid"

###

# obviously if the daemon doesn't exist we should stop now
if [ ! -x $DAEMON ]; then
        exit 0
fi

# grab an arbitrary config setting from nsca.cfg
get_config(){
        grep "^[[:space:]]*$1=" $CONF 2>/dev/null | tail | cut -d= -f2-
}

# if the pid_file is specified in the configuration file, nsca will
# take care of the pid handling for us.  if it isn't we should continue
# as we have before
PIDFILE=`get_config pid_file`
# if pidfile isn't set
if [ -z "$PIDFILE" ];  then
        # then this is the default PIDFILE
        PIDFILE="/var/run/nsca.pid"
        # run nsca in the foreground, and have s-s-d fork it for us
        OPTS="-f $OPTS"
        # and then this is how we call SSD
        SSD_STARTOPTS="--background --pidfile $PIDFILE --make-pidfile"
        SSD_STOPOPTS="--pidfile $PIDFILE"
else
        # but if pid_file is set, we don't have to do anything
        SSD_STARTOPTS="--pidfile $PIDFILE"
        SSD_STOPOPTS="--pidfile $PIDFILE"
fi

SSD_START="/sbin/start-stop-daemon -S $SSD_STARTOPTS --exec $DAEMON"
SSD_STOP="/sbin/start-stop-daemon -K $SSD_STOPOPTS --exec $DAEMON"

die(){
        echo $@
        exit 1
}

case "$1" in
start)
        echo -n "Starting $DESC: "
        if [ ! -d "/var/run/nagios" ]; then
```

```
                mkdir -p /var/run/nagios || die "ERROR: couldn't create
/var/run/nagios"
        fi
        $SSD_START -- $OPTS || die "ERROR: could not start $NAME."
        echo "$NAME."
;;
stop)
        echo -n "Stopping $DESC: "
        $SSD_STOP -- $OPTS || die "ERROR: could not stop $NAME."
        rm -f $PIDFILE
        echo "$NAME."
;;
reload|force-reload)
        echo -n "Reloading $DESC: "
        $SSD_STOP --signal HUP -- $OPTS || die "ERROR: could not reload $NAME."
        echo "$NAME."
;;
restart)
        $0 stop
        $0 start
;;
Esac
```

 Il sera lancé au démarrage de la machine :

```
# chmod +x /etc/init.d/nsca

# update-rc.d nsca defaults
```

8.6.2 – Installation de Nsca côté machine à superviser (serveur-nagios)

Pas besoin de refaire l'installation, vous l'avez déjà faite plus haut dans ce TP, étant donné que le serveur Nsca et le client Nsca sont sur le même serveur dans le cadre de ce TP. Il vous suffira juste de copier le binaire et le fichier de configuration (voir ci-dessous).

 - Copier le binaire et le fichier de configuration qui nous intéresse pour la partie cliente.

```
# cp src/send_nsca /usr/local/nagios/libexec

# cp sample-config/send_nsca.cfg /usr/local/nagios/etc/
```

 Quelques explications sur le fichier de configuration (send_nsca.cfg) :

```
# A METTRE LE MOT DE PASSE QUE VOUS DESIREZ
```

```
password=xxxxxxxx

# LA METHODE DE CRYPTAGE (FAITE VOTRE CHOIX)
encryption_method=1
```

Le mot de passe et la méthode de cryptage doivent être à l'identique que ceux spécifiés dans le nsca.cfg

8.6.3 – Fonctionnement côté machine à superviser (serveur-nagios)

Le principe est simple, **send_nsca** aura pour mission d'envoyer une commande externe au serveur nagios pour lui confirmer dans quel état il se trouve, donc vos scripts devront à la fin faire appel à **send_nsca**. Ce qui est le cas de notre script **rsyslog-nagios.sh**.

Comme NRPE, **send_nsca** est avant tout un moyen de transport pour soumettre les résultats de contrôles locaux au serveur Nagios; **send_nsca** contactant pour se faire le démon NSCA fonctionnant sur celui-ci. send_nsca possède les mêmes arguments et options ainsi que le même fichier de configuration sur toutes les plates-formes pour lequel il fonctionne.

```
-H <host_address>
L'adresse IP du serveur Nagios à contacter.

[-p port]
Le port distant à contacter (défaut 5667).

[-to to_sec]
Le seuil de dépassement de temps de connexion exprimé en secondes. La valeur par
défaut est 10.

[-d delim]
Le séparateur à utiliser pour présenter les données (tabulation par défaut).

[-c config_file]
Fichier de configuration à utiliser.

Selon que vous retournez au serveur un contrôle d'hôte ou de service, le format des
données varie légèrement.

Contrôles de services :

<code><host_name>[tab]<svc_description>[tab]<return_code>[tab]<plugin_output>[newli
ne]</code>

Contrôles d'hôtes :

<host_name>[tab]<return_code>[tab]<plugin_output>[newline]
```

Dans les deux cas, il faut présenter le nom de l'hôte ; le code retour et le message à afficher dans la console. Dans le cas du contrôle de service, il faut en plus envoyer la description du service à impacter dans Nagios.

Nous allons lancer cette commande à la main pour voir si la configuration est correcte et si le lien entre les deux serveurs fonctionne.

```
# echo "serveur-linux;ntp_critical;0;OK  - Soumission via send_nsca"
|/usr/local/nagios/libexec/send_nsca -H @ipserveurNagios -c
/usr/local/nagios/etc/send_nsca.cfg -d ";"
1 data packet(s) sent to host successfully.
```

Dans cet exemple d'appel, nous utilisons un simple echo dont la sortie est passée via un tube à l'entrée de **send_nsca**. Les données sont présentées avec un double point virgule comme séparateur grâce à l'argument -d et le fichier de configuration est précisé avec l'option -c. send_nsca répond en indiquant que le message a bien été transmis.

Lorsque le client NTP se synchronisera, le système de filtrage exécutera le script **rsyslog-nagios.sh** qui retournera un echo du même format que l'exemple ci-dessus. Il ne reste plus qu'à configurer le service Nagios correspondant.

8.7 – Définition du service Nagios

 C'est NSCA qui reçoit ce message sur le port d'écoute précisé et transforme ce message en commande externe Nagios. Pour que cette commande soit bien prise en compte par Nagios, il faut bien entendu avoir l'hôte et le service précisé présent dans les fichiers de configuration.

```
define service{
        use                             generic-service
        service_description             ntp_critical
        host_name                       serveur-linux
        active_checks_enabled           0
        passive_checks_enabled          1
        is_volatile                     1
        max_check_attempts              1
        check_freshness                 1
        freshness_threshold             4200
        check_command                   check_dummy!2!No messages in last 6
hours
}
```

Nous créons un fichier qui définit un service pour serveur-linux appelé ntp_critical. Sa particularité est d'être un service passif grâce aux directives active_check_enabled à 0 et passive_checks_enabled à 1.

- **max_check_attempts** est à 1 pour passer le service directement en état HARD.
- **check_freshness** à 1 indique que nous souhaitons utiliser la fonction de contrôle de fraicheur d'un service de nagios.
- **freshness_threshold** à 4200 indique que nous déclencherons ce contrôle 4200secondes soit 70 minutes après la réception du dernier contrôle.
- **Check_command** est la commande check_dummy, nous lui passons comme paramètre 2 pour état critique et le message qui s'affiche dans la console d'administration.

 En résumé : si le système de filtrage de logs ne voit pas passer la chaine de caractères définie, le script ne s'exécutera pas et donc nagios ne sera pas alerté avec le code retour 0 (OK). Nagios

considérera que le serveur NTP ne c'est pas bien synchronisé, il exécutera donc la commande check_dummy avec la valeur 2 en argument, c'est-à-dire CRITICAL.