Gestion des problèmes de configuration - Édition - Météo

Sommaire

Contexte

Simple Network Management Protocol (abrégé **SNMP**), est un protocole de communication qui permet aux administrateurs réseau de gérer les équipements du réseau, de superviser et de diagnostiquer des problèmes réseaux et matériels à distance.

Une requête SNMP est un datagramme UDP envoyé par le manager à destination du port 161 de l'agent. L'agent répond alors au manager avec la valeur demandée.

Les traps SNMP, elles sont émises depuis les agents SNMP vers une destination (un serveur de supervision par exemple), qui entendra ces requêtes. Pour les comprendre, ce serveur devra disposer de bases de données avec l'ensemble des informations (OID et Descripteurs) des constructeurs, ces bases sont appelées MIB. Les valeurs pourront alors être interprétées par le serveur de supervision. Ce procédé est souvent utilisé dans les routeurs pour par exemple, avertir qu'un lien vient de tomber sur l'une de ses interfaces. L'intérêt des traps SNMP est donc d'envoyer des « alertes » dès qu'une panne apparaît sans attendre que le serveur de supervision le détecte de lui-même pendant une vérification dans le cadre d'un monitoring actif.

Il se peut donc que vous souhaitiez paramétrer Shinken pour récupérer et interpréter ces traps, nous vous proposons deux moyens, via le module WS Reciever de Shinken, ou via le module Named Pipe (aussi utilisé de manière historique dans Nagios avec le fichier nagios.cmd, que nous allons utiliser via un fichier shinken.cmd).

Attention : les traps SNMP doivent être utilisés dans un réseau sécurisé, car comme nous l'avons dit, la communication s'effectue en UDP, ce qui peut être utilisé par des personnes malintentionnées pour faire du DDOS ou encore propager des fausses traps.

Recevoir des Statuts au Format Shinken de l'extérieur

Recevoir des statuts de l'extérieur se fait par l'intermédiaire du Receiver. Il faut accrocher **l'un des 2 modules** suivants sur le Receiver :

- **receiver-module-webservice** (réception des statuts via une API REST)
  - Permets de recevoir les statuts de n'importe quel outil de transformation de trap SNMP situé n'importe où.
  - **Mise en place du module**, cf Module receiver-module-webservice
  - Mise en place d'un script qui interprète les traps, cf Script d'interprétation des traps avec le module receiver-module-webservice

- **named-pipe** (réception via un fichier "passe plat" FIFO (shinken.cmd))
  - Son mode de fonctionnement impose que les outils de réception des traps soient sur l'ordinateur hébergeant le Receiver.
  - **Mise en place du module**, cf Module named-pipe
  - Mise en place d'un script qui interprète les traps, cf Script d'interprétation des traps avec le module named-pipe

Nous vous recommandons l'utilisation du **receiver-module-webservice** qui est le module que nous ferons évoluer dans le temps, car il est beaucoup moins limitatif dans son fonctionnement.

Réception des TRAPS SNMP

Pour traiter les tras SNMP venant des équipements à superviser, il faut un serveur SNMP capable de capturer les traps, ainsi que traducteur de trap vers Shinken

• SNMPD	SNMP est un protocole de communication qui permet aux administrateurs réseau de gérer les équipements du réseau, de superviser et de diagnostiquer des problèmes réseau et matériel à distance. SNMPD est le démon SNMP.
• SNMPTRAP	SNMPTRAP est le service permettant de récupérer les traps SNMP envoyés par les équipements.
• SNMPTT	(SNMP Trap Translator) est un logiciel permettant de capturer et de traduire de manière compréhensible les messages *trap* remontés par les agents SNMP.

Installation

Des paquets SNMPD, SNMPTRAPD

Voici les étapes à suivre :

- Installation des paquets et dépendances :

```
yum install net-snmp net-snmp-utils net-snmp-perl perl-Sys-Syslog
```

Note : le paquet net-snmp est pour la partie serveur, et net-snmpd-utils est pour la partie cliente afin de diagnostiquer plus rapidement des problèmes SNMP sur vos serveurs. Mais ce n'est pas obligatoire. Les paquets net-snmp-perl et snmptt permettront de translater les traps.

- Pour activer le démarrage des services SNMP au démarrage du serveur avec la commande suivante :

```
chkconfig --level 3 snmpd on;chkconfig --level 3 snmptrapd on
```

Démarrer les services snmpd et snmptrapd :

```
service snmpd start;service snmptrapd start
```

Vous pouvez alors tester de passer une requête via les outils clients SNMP, sur son propre serveur pour vérifier la bonne communication :

```
snmpwalk -v 1 -c public -O e 127.0.0.1
```

La commande doit retourner une réponse comme par exemple :

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux Shinken 2.6.32-504.16.2.el6.x86_64 #1 SMP Wed Apr 22 06:48:29 UTC
2015 x86_64
SNMPv2-MIB::sysObjectID.0 = OID: NET-SNMP-MIB::netSnmpAgentOIDs.10
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (31359) 0:05:13.59
SNMPv2-MIB::sysContact.0 = STRING: Root <root@localhost> (configure /etc/snmp/snmp.local.conf)
SNMPv2-MIB::sysName.0 = STRING: shinken
SNMPv2-MIB::sysLocation.0 = STRING: Unknown (edit /etc/snmp/snmpd.conf)
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (64) 0:00:00.64
SNMPv2-MIB::sysORID.1 = OID: SNMP-MPD-MIB::snmpMPDMIBObjects.3.1.1
SNMPv2-MIB::sysORID.2 = OID: SNMP-USER-BASED-SM-MIB::usmMIBCompliance
SNMPv2-MIB::sysORID.3 = OID: SNMP-FRAMEWORK-MIB::snmpFrameworkMIBCompliance
SNMPv2-MIB::sysORID.4 = OID: SNMPv2-MIB::snmpMIB
SNMPv2-MIB::sysORID.5 = OID: TCP-MIB::tcpMIB
SNMPv2-MIB::sysORID.6 = OID: IP-MIB::ip
SNMPv2-MIB::sysORID.7 = OID: UDP-MIB::udpMIB
SNMPv2-MIB::sysORID.8 = OID: SNMP-VIEW-BASED-ACM-MIB::vacmBasicGroup
(...)
```

De SNMPTT

Mise en place

Le téléchargement se fait directement sur le site de l'éditeur. Vous pouvez le télécharger aussi ici -> snmptt_1.4.2.tgz. Il n'est actuellement pas disponible à l'installation avec yum.

- Copier l'archive sur votre serveur et décompressez-là puis allez dans le dossier créé :

```
tar zxvf snmptt_1.4.2.tgz
cd snmptt_1.4.2
```

- Exécutez les commandes suivantes pour l'installation de SNMPTT :

```
cp snmptt /usr/sbin/
cp snmptthandler /usr/sbin/
cp snmpttconvert /usr/sbin/
cp snmpttconvertmib /usr/sbin/
cp snmptt.ini /etc/snmp/
cp examples/snmptt.conf.generic /etc/snmp/snmptt.conf
mkdir /var/log/snmptt/
```

- Mettre en commentaire le paramètre daemon_uid du fichier /etc/snmp/snmptt.ini

```
#daemon_uid=snmptt
```

- Ajouter la ligne suivant au fichier /etc/snmp/snmptrapd.conf :

```
traphandle default /usr/sbin/snmptt
```

- Continuez avec :

```
mkdir /var/spool/snmptt/
chmod 777 /var/spool/snmptt/
cp snmptt-init.d /etc/rc.d/init.d/snmptt
chkconfig --add snmptt
chkconfig --level 2345 snmptt on
```

- Démarrer snmptt :

```
service snmptt start
```

Configuration

Maintenant que l'on est capable de recevoir des traps, il faut les interpréter pour dialoguer ensuite avec Shinken :

- On active le debug dans le fichier **/etc/snmp/snmptt.ini**, histoire de voir les traps que l'on a reçus, mais pas interprétés.
  Si vous avez mal configuré quelque chose, elles arriveront dans le fichier **/var/log/snmptt/snmpttunknown.log**.

```
unknown_trap_log_enable = 1
```

- On ajoute ensuite les lignes suivantes dans la configuration sous **/etc/snmp/snmptrapd.conf** :

```
disableAuthorization yes
donotlogtraps  no
```

- Ce qui veut dire :
  - disableAuthorization : on accepte toutes les interruptions (traps)
  - donotlogtraps : on désactive le log des interruptions reçues (c'est SNMPTT qui s'en chargera)
  - traphandle default /usr/sbin/snmptt : on traite de la même façon toutes les interruptions : avec SNMPTT
- Le service snmptrapd doit être modifié pour ne pas traduire les OID, mais les laisser sous forme numérique.
  - C'est SNMPTT qui se chargera de cette traduction.
  - Sous CentOS, il faut éditer le fichier **/etc/sysconfig/snmptrapd** et ajouter l'option comme ceci:

```
OPTIONS="-Lsd -p /var/run/snmptrapd.pid"
```

- Il faut activer également le processus de capture de trap nommé snmptrapd. Ceci se fait dans le même fichier :

```
export MIBS=/usr/share/mibs
TRAPDRUN=yes
```

- Pour avoir le droit d'exécuter un script à la réception d'une trap SNMP, il faut ajouter une permission à SNMP dans SELinux :
  Doit être dans la procédure d'install ?

```
semanage permissive -a snmpd_t
```

- On relance les deux services pour prendre en compte les modifications :

```
service snmpd restart
service snmptt restart
```

Compilation de MIB

SNMPTT a besoin de compiler les MIBs du format TXT vers un fichier de configuration. Cette compilation effectue l'association OID/action à effectuer (information inscrite dans le fichier MIB).

- Les MIBs de CentOS se trouvent dans le dossier **/usr/share/snmp/mibs/**.
- Si vous voulez travailler avec une MIB particulière (routeur CISCO ou autre équipement), il vous faudra l'importer sur le système afin de la compiler.

Pour chaque MIB, il faut compiler à l'aide de la syntaxe suivante :

```
snmpttconvertmib --in=<fichier MIB> \
--out=/etc/snmp/snmptt.conf.<equipement> \
--exec='/var/lib/shinken-user/libexec/submit_check_result_to_receiver $r TRAP 2'
```

Important

C'est le script /var/lib/shinken-user/libexec/submit_check_result_to_receiver qui va faire le lien avec shinken

C'est la commande de vous avez activé dans la procédure mise en place du module de réception sur le Receiver:

- Script d'interprétation des traps avec le module receiver-module-webservice
- Module named-pipe

Résumé du fonctionnement

Vous avez plus qu'à importer vos MIBs et faire la liaison avec Shinken de la même façon que dans cet exemple.

Pour résumer, nous avons bien la chaîne suivante:

Résumé

SNMPD (pour l'écoute des Traps) **–> SNMPTT** (pour la translation des Traps) **–> Script submit_check_result_to_receiver** (pour le passage au format Shinken) –> **Module de réception sur le Receiver** (pour l'envoi de l'état du check de l'hôte dans Shinken)

Vérifier le bon fonctionnement (Test avec une MIB perso)

Création d'un check passif

Dans l'interface de configuration, il faut créer le check à associer à un hôte, en passif, non actif, et volatile, car nous souhaitons recevoir une notification dès un changement d'état. L'expiration de l'état de fraîcheur (freshness_threshold) doit également être paramétré.

Voici un exemple avec la syntaxe d'un fichier de configuration, modèle de check dédié au modèle d'hôte "TRAP-modele" :

```
define service{
        service_description     TRAP
        check_command           check-host-alive
        host_name               TRAP-modele
                is_volatile             1
        passive_checks_enabled  1
                active_checks_enabled          0
                check_freshness                        1
                freshness_threshold               300
        register                0
        check_interval              1
        retry_interval              1
}

define host{
        name                                TRAP-modele
        register                        0
}
```

Appliquer le modèle d'hôte "TRAP-modele" à un hôte accessible sur le réseau (le paramètre "adresse" doit être rempli), par exemple un hôte "test-trap".

⚠ **Rappels sur le fonctionnement des checks passifs**

La configuration de check présentée ci-dessus est celle d'un check passif. Le statut d'un check de ce type va être mis à jour manuellement via la réception de traps SNMP.

Si aucun statut n'est reçu de manière passive pour ce check au bout de 5min, Shinken déclenche une vérification manuelle pour éviter d'avoir un statut trop ancien et non représentatif de l'état de l'élément représenté par le check. Ce comportement est configuré via les options "check_freshness" (pour l'activation de ce comportement), "check_command" (pour la commande de vérification lancée par Shinken) et "freshness_threshold" (en seconde, 300 pour 5min).

Par contre, ce comportement peut entraîner des changements de statuts du check si l'intervalle d'envoi des statuts vers le check (traps SNMP dans notre exemple) est supérieur à l'intervalle défini par l'option "freshness_threshold". Il faut donc avoir conscience de ce comportement et régler la valeur de l'option "freshness_threshold" ou bien désactiver cette fonctionnalité pour ce check ("check_freshness 0").

Envoi d'une trap via une MIB perso sur l'hôte Shinken uniquement

Exemple dans le cas ou vos traps SNMP n'ont besoin d'arriver que sur votre hôte Shinken

On va créer une **MIB de test** pour pouvoir tester toute la chaîne depuis la capture de la trap jusqu'au traitement de celle ci par Shinken :

- Paramétrer le nom du serveur dans le fichier /etc/snmp/snmptt.ini

```
snmptt_system_name =
```

- On créé notre MIB dans un fichier sous **/usr/share/snmp/mibs/TRAP-TEST-MIB** avec le contenu suivant :

```
TRAP-TEST-MIB DEFINITIONS ::= BEGIN
IMPORTS ucdExperimental FROM UCD-SNMP-MIB;
demotraps OBJECT IDENTIFIER ::= { ucdExperimental 990 }
demo-trap TRAP-TYPE
STATUS current
ENTERPRISE demotraps
VARIABLES { sysLocation }
DESCRIPTION "Trap received !"
::= 17
END
```

- On export cette MIB :

```
export MIBS=+/usr/share/snmp/mibs/TRAP-TEST-MIB
```

- **On la compile en précisant notre plugin submit_check_result_to_receiver, c'est donc ici que ce fait la jonction avec Shinken :**

```
snmpttconvertmib --in=/usr/share/snmp/mibs/TRAP-TEST-MIB \
--out=/etc/snmp/snmptt.conf.test \
--exec='/var/lib/shinken-user/libexec/submit_check_result_to_receiver $r TRAP 2'
```

- Ce qui doit générer un fichier de cette forme :

```
EVENT demo-trap .1.3.6.1.4.1.2021.13.990.0.17 "Status Events" Normal
FORMAT Trap received ! $*
EXEC /var/lib/shinken-user/libexec/submit_check_result_to_receiver $H TRAP 2 "Trap received ! $*"
SDESC
Trap received !
Variables:
  1: sysLocation
EDESC
```

- $H correspond à la variable "snmptt_system_name"
- $* correspond toutes les variables textuelles défini lors de l'envoi de la trap ("It's a trap" plus bas dans notre exemple)
- On fait prendre en compte ce fichier à la configuration globale de SNMPTT à la fin du fichier **/etc/snmp/snmptt.ini** en rajoutant notre fichier généré snmptt.conf.test :

```
snmptt_conf_files = <<END
/etc/snmp/snmptt.conf
/etc/snmp/snmptt.conf.test
END
```

- On relance les services :

```
service snmptt restart
service snmpd restart
service snmptrapd restart
```

- On lance une trap de test à la main :

```
snmptrap -v 1 -c public 127.0.0.1 TRAP-TEST-MIB::demotraps localhost 6 17 '' SNMPv2-MIB::sysLocation.0 s
"It's a trap"
```

- Les logs de snmptt doivent remonter la trap :

```
tail /var/log/snmptt/snmptt.log
Thu Mar 18 16:57:38 2021 .1.3.6.1.4.1.2021.13.990.0.17 Normal "Status Events" localhost - Trap received !
It's a trap
```

- Même chose pour les log messages du système :

```
tail /var/log/messages
Mar 18 16:57:38 shinken281 snmptrapd[6539]: 2021-03-18 16:57:38 localhost [127.0.0.1] (via UDP: [127.0.0.1]:
52791->[127.0.0.1]:162) TRAP, SNMP v1, community public#012#011UCD-SNMP-MIB::ucdEx
perimental.990 Enterprise Specific Trap (17) Uptime: 6:54:17.07#012#011SNMPv2-MIB::sysLocation.0 = STRING:
It's a trap
Mar 18 16:57:38 shinken281 snmptt[17098]: .1.3.6.1.4.1.2021.13.990.0.17 Normal "Status Events" localhost -
Trap received ! It's a trap
```

- Le service passe alors en critique sur Shinken et vous recevez une notification.

Au bout de une minute, comme défini dans le fichier de configuration sur Shinken, le service passe de nouveau au vert en statut OK.

Envoi d'une trap via une MIB perso sur un ou plusieurs hôtes différents

Exemple dans le cas ou vous avez plusieurs hôtes différents sur lequel envoyer une trap SNMP

On va créer une **MIB de test** pour pouvoir tester toute la chaîne depuis la capture de la trap jusqu'au traitement de celle ci par Shinken :

- On créé notre MIB dans un fichier sous **/usr/share/snmp/mibs/TRAP-TEST-MIB** avec le contenu suivant :

```
TRAP-TEST-MIB DEFINITIONS ::= BEGIN
IMPORTS ucdExperimental FROM UCD-SNMP-MIB;
demotraps OBJECT IDENTIFIER ::= { ucdExperimental 990 }
demo-trap TRAP-TYPE
STATUS current
ENTERPRISE demotraps
VARIABLES { sysLocation }
DESCRIPTION "Trap received !"
::= 17
END
```

- On export cette MIB :

```
export MIBS=+/usr/share/snmp/mibs/TRAP-TEST-MIB
```

- **On la compile en précisant notre plugin submit_check_result_to_receiver, c'est donc ici que ce fait la jonction avec Shinken :**

```
snmpttconvertmib --in=/usr/share/snmp/mibs/TRAP-TEST-MIB \
--out=/etc/snmp/snmptt.conf.test \
--exec='/var/lib/shinken-user/libexec/submit_check_result_to_receiver $r TRAP 2'
```

- Ce qui doit générer un fichier de cette forme :

```
EVENT demo-trap .1.3.6.1.4.1.2021.13.990.0.17 "Status Events" Normal
FORMAT Trap received ! $*
EXEC /var/lib/shinken-user/libexec/submit_check_result_to_receiver $1 TRAP 2 "Trap received ! $2 $1"
SDESC
Trap received !
Variables:
  1: sysLocation
EDESC
```

- $1 correspond au nom de l'hôte a qui la trap est envoyée
- $* correspond à la variable textuelle défini lors de l'envoi de la trap ("It's a trap" plus bas dans notre exemple)
- On fait prendre en compte ce fichier à la configuration globale de SNMPTT à la fin du fichier **/etc/snmp/snmptt.ini** en rajoutant notre fichier généré snmptt.conf.test :

```
snmptt_conf_files = <<END
/etc/snmp/snmptt.conf
/etc/snmp/snmptt.conf.test
END
```

- On relance les services :

```
service snmptt restart
service snmpd restart
service snmptrapd restart
```

- Modifier le fichier /etc/snmp/snmptt.conf.test :

```
EVENT demo-trap .1.3.6.1.4.1.2021.13.990.0.17 "Status Events" Normal
FORMAT Trap received ! $*
EXEC /var/lib/shinken-user/libexec/submit_check_result_to_receiver $1 TRAP 2 "Trap received ! $2 $1"
SDESC
Trap received !
Variables:
  1: sysLocation
EDESC
```

- Lancement d'une trap de test à la main :

```
snmptrap -v 1 -c public 127.0.0.1 TRAP-TEST-MIB::demotraps localhost 6 17 '' SNMPv2-MIB::sysLocation.0 s
"NOM_DE_L'HÔTE" SNMPv2-MIB::sysLocation.0 s "It's a trap"
```

- Cette trap ne sera pas envoyé sur le serveur Shinken mais sur l'hôte désiré