Gestion des traps SNMP

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 envoyée 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 Arbiter 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 utilisées 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 Status au Format Shinken de l'extérieur

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

- WSArbiter (réception des statuts via API REST)
  - Permets de recevoir les status de n'importe quel outil de transformation de trap SNMP situé n'importe ou.
  - **Mise en place du module**, cf Module ws-arbiter (mise en place)
- NamedPipe (réception via un fichier "passe plat" FIFO)
  - Son mode de fonctionnement impose que les outils de réception des traps soit sur l'ordinateur hébergeant le Receiver.
  - **Mise en place du module**, cf Module named-pipe (mise en place)

Important

Nous vous recommandons l'utilisation du WSArbiter 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 traps 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

> **Attention**
>
> Une utilisation des traps SNMP nécessite de désactiver la vérification active sur le check intéressé et/ou sur l'hôte en fonction de l'élément visé.
>
> Pour ce faire, il faut se rendre dans l'interface de configuration sur la page de votre hôte ou votre check, et dans la partie supervision, mettre a faut la ligne ***Actif activé*** à Faux.

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 traduire les traps.

- Pour activer les services SNMP au démarrage du serveur, il faut exécuter 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 et passer une requête, via les outils clients SNMP, sur le serveur pour vérifier la bonne communication :

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

La commande doit retourner une réponse de la même forme que celle-ci :

```
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
(...)
```

Installation de SNMPTT

Mise en place

- Le téléchargement du programme se fait directement avec yum :

```
yum install snmptt
```

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

```
#daemon_uid = snmptt
```

- Démarrer snmptt :

```
service snmptt start
```

Configuration

Maintenant que le serveur est capable de recevoir des traps, il faut les interpréter pour envoyer des statuts à Shinken :

- On édite le fichier de configuration **/etc/snmp/snmptt.ini**
  - On active le niveau de log debug afin de voir les traps reçus mais non interprétés. S'il y a une erreur dans la configuration, les traps non interprétés seront écrits dans le fichier : **/var/log/snmptt/snmpttunknown.log**.

```
unknown_trap_log_enable = 1
```

  - L'activation du module Perl net-snmp permettra l'interprétation des OIDs :

```
net_snmp_perl_enable = 1
```

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

```
disableAuthorization yes
traphandle default /usr/sbin/snmptthandler
```

  - Cela permet :
    - **disableAuthorization** : on accepte toutes les interruptions (traps)
    - **traphandle default /usr/sbin/snmptt** : activation de snmptthandler

- Dans le fichier de configuration **/etc/sysconfig/snmptrapd**

- Le service **snmptrapd** doit être modifié pour ne pas traduire les OID, et les laisser sous forme numérique. **SNMPTT** se chargera alors de la traduction. Ajouter l'option comme ceci :

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

- Il faut également activer le processus de capture de trap nommé **snmptrapd** :

```
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 :

```
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 d'une des syntaxes suivantes :

Cette syntaxe permet simplement de compiler une seule MIB :

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

Cette syntaxe permet de compiler toutes les MIBs d'un répertoire :

```
for i in *; do snmpttconvertmib --in=$i --out=snmptt.conf.test --exec='/var/lib/shinken-user/libexec
/submit_check_result_to_receiver $r TRAP 2'; done
```

Il est aussi possible de reprendre cette syntaxe, et mettre par exemple CISCO juste devant l'étoile, comme ceci :

```
for i in CISCO*; do snmpttconvertmib --in=$i --out=snmptt.conf.test --exec='/var/lib/shinken-user/libexec
/submit_check_result_to_receiver $r TRAP 2'; done
```

Ici, sera converti toutes les MIBs du répertoire courant commençant par CISCO.

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 que vous avez activée dans la procédure mise en place du module de réception sur le Receiver. (*Voir la page* Module named-pipe (mise en place))

Problème récurrent

Attention, lors de la compilation de la MIB, il est possible que vous rencontriez un problème similaire a celui-ci :

```
$ snmpttconvertmib --in=/usr/share/snmp/mibs/CISCO-OSCP-MIB.mib --out=/etc/snmp/test.txt --exec='/var/lib
/shinken-user/libexec/submit_check_result_to_receiver $r TRAP 2'
exec: /var/lib/shinken-user/libexec/submit_check_result_to_receiver $r TRAP 2


*****  Processing MIB file *****

snmptranslate version: NET-SNMP version: 5.7.2
severity: Normal

File to load is:        /usr/share/snmp/mibs/CISCO-OSCP-MIB.mib
File to APPEND TO:      /etc/snmp/test.txt

MIBS environment var:   /usr/share/snmp/mibs/CISCO-OSCP-MIB.mib
mib name: CISCO-OSCP-MIB


Processing MIB:         CISCO-OSCP-MIB
#
skipping a TRAP-TYPE / NOTIFICATION-TYPE line - probably an import line.
#
Line: 677
NOTIFICATION-TYPE: coscpNotifyTransDown
Variables: coscpLinkTransDown
Enterprise: ciscoOscpNotificationsPrefix
Looking up via snmptranslate: CISCO-OSCP-MIB::coscpNotifyTransDown
MIB search path: /root/.snmp/mibs:/usr/share/snmp/mibs
Cannot find module (CISCO-SMI): At line 31 in /usr/share/snmp/mibs/CISCO-OSCP-MIB.mib
Did not find 'ciscoMgmt' in module #-1 (/usr/share/snmp/mibs/CISCO-OSCP-MIB.mib)
Unlinked OID in CISCO-OSCP-MIB: ciscoOscpMIB ::= { ciscoMgmt 202 }
Undefined identifier: ciscoMgmt near line 34 of /usr/share/snmp/mibs/CISCO-OSCP-MIB.mib
Cannot adopt OID in CISCO-OSCP-MIB: ciscoOscpMIBGroups ::= { ciscoOscpMIBConformance 2 }
```

Pour résoudre le problème, il faut regarder la ligne suivante :

```
Cannot find module (CISCO-SMI): At line 31 in /usr/share/snmp/mibs/CISCO-OSCP-MIB.mib
```

On comprend donc que snmpttconvertmib n'a pas accès au module CISCO-SMI.

La raison de ce problème est que la MIB que vous essayez de compiler a des dépendances que l'on peut retrouver dans le fichier même de la MIB que vous voulez compiler.

Si vous regardez votre fichier, vers le début de celui-ci, vous pourrez retrouver un bloc comme celui-ci :

```
IMPORTS
        MODULE-IDENTITY,
        OBJECT-TYPE,
        NOTIFICATION-TYPE,
        Counter32,
        Gauge32,
        Unsigned32
                                                FROM SNMPv2-SMI
        TEXTUAL-CONVENTION,
        TruthValue,
        RowStatus
                                                FROM SNMPv2-TC
        InterfaceIndex
                                                FROM IF-MIB
        MODULE-COMPLIANCE,
        OBJECT-GROUP,
        NOTIFICATION-GROUP
                                                FROM SNMPv2-CONF
        ciscoMgmt
                                                FROM CISCO-SMI
        ;
```

Il indique toutes les dépendances nécessaires pour compiler votre MIB.

Dans cet exemple, on remarque donc que les MIBs nécessaires pour la compiler sont :

- **SNMPv2-SMI**
- **SNMPv2-TC**
- **IF-MIB, SNMPv2-CONF**
- **CISCO-SMI.**

Nous pouvons remarquer que CISCO-SMI est aussi le module indiqué dans le message d'erreur.

Pour régler le problème, il faut télécharger la MIB manquante. On pourra par exemple se diriger sur ce site : http://www.circitor.fr/Mibs/Mibs.php qui répertorie une large variété de MIBs différentes.

Une fois la MIB téléchargée, il suffit de la mettre dans le dossier **/usr/share/snmp/mibs** du serveur de supervision, avec la MIB à convertir.

Relancer la commande de compilation de la MIB, qui donnera un message de retour ressemblant à celui-ci :

```
$ snmpttconvertmib --in=/usr/share/snmp/mibs/CISCO-OSCP-MIB.mib --out=/etc/snmp/test.txt --exec='/var/lib
/shinken-user/libexec/submit_check_result_to_receiver $r TRAP 2'
exec: /var/lib/shinken-user/libexec/submit_check_result_to_receiver $r TRAP 2


*****  Processing MIB file *****

snmptranslate version: NET-SNMP version: 5.7.2
severity: Normal

File to load is:        /usr/share/snmp/mibs/CISCO-OSCP-MIB.mib
File to APPEND TO:      /etc/snmp/test.txt

MIBS environment var:   /usr/share/snmp/mibs/CISCO-OSCP-MIB.mib
mib name: CISCO-OSCP-MIB


Processing MIB:         CISCO-OSCP-MIB
#
skipping a TRAP-TYPE / NOTIFICATION-TYPE line - probably an import line.
#
Line: 677
NOTIFICATION-TYPE: coscpNotifyTransDown
Variables: coscpLinkTransDown
Enterprise: ciscoOscpNotificationsPrefix
Looking up via snmptranslate: CISCO-OSCP-MIB::coscpNotifyTransDown
OID: .1.3.6.1.4.1.9.9.202.2.0.1
```

Résumé du fonctionnement

Il ne reste plus qu'à importer les MIBs et faire la liaison avec Shinken de la même façon que dans l'exemple.

Pour résumer, la chaîne est la suivante 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 factice)

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. Le seuil d'expiration des états reçus des outils externes (*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 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 :

- Vérification que l'état reçu des outils externes ne soit pas expiré (*check_freshness*) : pour l'activation de ce comportement,
- Vivant (Commande de vérification) (*check_command*) (pour la commande de vérification lancée par Shinken) et
- Le seuil d'expiration des états reçus des outils externes (*freshness_threshold*) : en seconde, 300 pour 5min.

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 e seuil d'expiration des états reçus des outils externes (*freshness_threshold*) **.**

Il faut donc régler la valeur du seuil d'expiration des états reçus des outils externes (*freshness_threshold*) ou bien désactiver cette fonctionnalité pour ce check en choisissant la valeur 0 pour le seuil.

Envoi d'un trap via une MIB factice

On va créer une **MIB factice** pour pouvoir tester toute la chaîne depuis la capture du trap jusqu'au traitement par Shinken :

- On crée notre MIB dans le fichier **/usr/share/snmp/mibs/UCD-TRAP-TEST-MIB** avec le contenu suivant :

```
UCD-TRAP-TEST-MIB DEFINITIONS ::= BEGIN
      IMPORTS ucdExperimental FROM UCD-SNMP-MIB;

demotraps OBJECT IDENTIFIER ::= { ucdExperimental 990 }

demoTrap TRAP-TYPE
      ENTERPRISE demotraps
      VARIABLES { sysLocation }
      DESCRIPTION "It's a trap !"
      ::= 17

END
```

- On exporte cette MIB :

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

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

```
snmpttconvertmib --in=/usr/share/snmp/mibs/UCD-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 :

```
MIB: UCD-TRAP-TEST-MIB (file:/usr/share/snmp/mibs/UCD-TRAP-TEST-MIB) converted on Tue Jan 31 15:03:39 2023
using snmpttconvertmib v1.4.2
#
#
#
EVENT demoTrap .1.3.6.1.4.1.2021.13.990.0.17 "Status Events" Normal
FORMAT It's a trap ! $*
EXEC /var/lib/shinken-user/libexec/submit_check_result_to_receiver $r TRAP 2 "It's a trap ! $*"
SDESC
It's a trap !
Variables:
  1: sysLocation
EDESC
```

- $r correspond au nom de l'hôte à qui le trap est envoyé
- $* correspond à la variable textuelle défini lors de l'envoi du 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'un trap de test manuellement :

```
snmptrap -v 1 -c public 127.0.0.1 UCD-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"
```

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

Le fichier de logs **/var/log/snmptt/snmptt.log** permettra de vérifier que le trap a bien été reçu :

```
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
```

Ou encore dans le fichier **/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 check associé à l'hôte défini plus haut passe alors en critique, ce qui permet de confirmer que le trap fonctionne.