Collecter les journaux d'instabilité du protocole de routage IOS-XE avec Python

Table des matières

Introduction

Ce document décrit comment configurer les scripts Python pour collecter les journaux OSPF, EIGRP et IS-IS lorsque les protocoles sont instables.

Conditions préalables

Exigences

Cisco vous recommande de vous familiariser avec les rubriques répertoriées :

- Configuration de l'hébergement des applications
- OSPF
- EIGRP
- IS-IS
- éditeur vi

Composants utilisés

Les informations contenues dans ce document sont basées sur la version 17 du logiciel Cisco IOS XE.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. Si votre réseau est en ligne, assurez-vous de bien comprendre l’incidence possible des commandes.

Remarque : Ce document n'aborde pas les détails de la capture. Vous trouverez plus d'informations dans les liens référencés.

Configurer

Configurations

Lorsque vous ouvrez un dossier TAC, il est très important de collecter des informations pertinentes pour gagner du temps. Parfois, l'indice d'une panne se trouve dans certaines sorties de base que vous pouvez obtenir du périphérique. Dans ce document, vous avez des exemples de la façon d'utiliser les scripts Python pour obtenir ces données. Trois protocoles sont pris en compte : OSPF, EIGRP et IS-IS.

Étape 1. La première chose à faire est de configurer et d’activer Guestshell.

```
Router(config)#iox
Router(config)#interface VirtualPortGroup 0
Router(config-if)#ip address 192.0.2.1 255.255.255.252
Router(config-if)#exit
Router(config)#
Router(config)#app-hosting appid guestshell
Router(config-app-hosting)#app-vnic gateway0 virtualportgroup 0 guest-interface 0
Router(config-app-hosting-gateway0)#guest-ipaddress 192.0.2.2 netmask 255.255.255.252
Router(config-app-hosting)#app-default-gateway 192.0.2.1 guest-interface 0
Router(config)#end
```

Dans cette configuration, il y a trois étapes importantes :

1. Activez le service IOX. Cette opération est nécessaire pour activer Guestshell.

2. Configurez le VirtualPortGroup qui agit comme passerelle par défaut pour la passerelle par défaut de Guestshell.

3. Configurez l'hébergement d'applications pour Guestshell. Les configurations vous indiquent où

le VirtualPortGroup entre en jeu.

Étape 2. Ensuite, vous devez activer guestshell à partir du mode privilégié.

```
Router#guestshell enable
Interface will be selected if configured in app-hosting
Please wait for completion
guestshell installed successfully
Current state is: DEPLOYED
guestshell activated successfully
Current state is: ACTIVATED
guestshell started successfully
Current state is: RUNNING
Guestshell enabled successfully

Router#
*Jun 15 21:31:31.499: %IM-6-IOX_INST_INFO: R0/0: ioxman: IOX SERVICE guestshell LOG: Guestshell is up a
```

Si tout est correctement configuré, vous devez voir le journal dans l'exemple précédent.

Étape 3. Maintenant, vous êtes prêt à configurer les scripts python. Exécutez la commande guestshell en mode privilégié. L'invite s'affiche comme dans l'exemple suivant :

```
Router#guestshell
[guestshell@guestshell ~]$
```

Étape 4. Créez un fichier avec vi editor et configurez les scripts en fonction des protocoles que vous avez activés.

```
[guestshell@guestshell ~]$ vi ospf.py
```

Cette fenêtre s'affiche

```
~
~
~
~
~
~
~
"ospf.py" 0L, 0C
```

Étape 5. Appuyez sur "i" pour insérer du texte. Collez le script, puis appuyez sur "esc", puis entrez les caractères : wq

```
~
from cli import cli
from time import sleep

cli("enable")
cli("debug ip ospf hello")
cli("debug ip ospf adj")
cli("show ip ospf interface | append bootflash:Router-ospf-logs.txt")
cli("show ip ospf neighbor | append bootflash:Router-ospf-logs.txt")
cli("show interfaces | append bootflash:Router-ospf-logs.txt")
cli("show logging | append bootflash:Router-ospf-logs.txt")
cli("show tech | append bootflash:Router-showtech.txt")
sleep(30)
cli("undebug all")
~
~
~
~
"ospf.py" [New] 14L, 458C written
[guestshell@guestshell ~]$
```

Quittez le shell d'invité avec la commande exit.

Vérifier

Testez le script. Quittez le shell d'invité avec la commande exit. Ensuite, exécutez guestshell run python3 ospf.py

```
F340.20.09-8500-1#guestshell run python3 ospf.py
```

Voici les scripts pour les trois protocoles ; OSPF, EIGRP et IS-IS.

OSPF

```
from cli import cli
from time import sleep

cli("enable")
cli("debug ip ospf hello")
cli("debug ip ospf adj")
cli("show ip ospf interface | append bootflash:Router-ospf-logs.txt")
cli("show ip ospf neighbor | append bootflash:Router-ospf-logs.txt")
```

```
cli("show interfaces | append bootflash:Router-ospf-logs.txt")
cli("show logging | append bootflash:Router-ospf-logs.txt")
cli("show tech | append bootflash:Router-showtech.txt")
sleep(30)
cli("undebug all")
```

EIGRP

```
from cli import cli
from time import sleep

cli("enable")
cli("debug eigrp packet")
cli("show ip eigrp neighbor | append bootflash:Router-eigrp-logs.txt")
cli("show ip eigrp interface | append bootflash:Router-eigrp-logs.txt")
cli("show interfaces | append bootflash:Router-eigrp-logs.txt")
cli("show logging | append bootflash:Router-eigrp-logs.txt")
cli("show tech | append bootflash:Router-showtech.txt")
sleep(30)
cli("undebug all")
```

IS-IS

```
from cli import cli
from time import sleep

cli("enable")
cli("debug isis adj-packet")
cli("show isis neighbor detail | append bootflash:Router-isis-logs.txt")
cli("show clns neighbor detail | append bootflash:Router-isis-logs.txt")
cli("show clns interface | append bootflash:Router-isis-logs.txt")
cli("show interfaces | append bootflash:Router-isis-logs.txt")
cli("show logging | append bootflash:Router-isis-logs.txt")
cli("show tech | append bootflash:Router-showtech.txt")
sleep(30)
cli("undebug all")
```

Vous pouvez automatiser la collecte des journaux avec des scripts EEM qui exécutent les scripts Python après l'observation des modèles syslog. Dans la section suivante, vous avez les scripts EEM que vous pouvez configurer avec les scripts python pour accomplir cette tâche.

OSPF

```
event manager applet ospf-flap authorization bypass
event syslog pattern "%OSPF-5-ADJCHG:.*from FULL to DOWN" maxrun 120 ratelimit 120
action 010 cli command "enable"
```

```
action 020 cli command "guestshell run python3 ospf.py"
action 030 exit
```

EIGRP

```
event manager applet eirgp-flap authorization bypass
event syslog pattern "%DUAL-5-NBRCHANGE: EIGRP.*Neighbor.*is down" maxrun 120 ratelimit 120
action 010 cli command "enable"
action 020 cli command "guestshell run python3 eigrp.py"
action 030 exit
```

IS-IS

```
event manager applet isis-flap authorization bypass
event syslog pattern "%CLNS-5-ADJCHANGE: ISIS: Adjacency to.*Down" maxrun 120 ratelimit 120
action 010 cli command "enable"
action 020 cli command "guestshell run python3 isis.py"
action 030 exit
```

Remarque : Les commandes collectées dans ces scripts fournissent des informations initiales de base. Lorsque vous ouvrez un dossier TAC, les ingénieurs TAC peuvent vous demander des informations supplémentaires pour approfondir vos recherches si nécessaire.

Liens de référence

- Guestshell
- API Python

À propos de cette traduction
Cisco a traduit ce document en traduction automatisée vérifiée par une personne dans le cadre d'un service mondial permettant à nos utilisateurs d'obtenir le contenu d'assistance dans leur propre langue.

Il convient cependant de noter que même la meilleure traduction automatisée ne sera pas aussi précise que celle fournie par un traducteur professionnel.