Recopilación de Registros Inestables del Protocolo de Ruteo IOS-XE con Python

Contenido

Introducción

Este documento describe cómo configurar Python Scripts para recopilar registros OSPF, EIGRP e IS-IS cuando los protocolos se inestabilizan.

Prerequisites

Requirements

Cisco recomienda que esté familiarizado con los temas que se indican a continuación:

- Configuración de alojamiento de aplicaciones
- OSPF
- EIGRP
- IS-IS
- editor vi

Componentes Utilizados

La información de este documento se basa en la versión 17 del software Cisco IOS XE.

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

Nota: Este documento no profundiza en los detalles de la captura. Puede encontrar más información en los enlaces de referencia.

Configurar

Configuraciones

Al abrir un caso de TAC, es muy importante recopilar información relevante para ahorrar tiempo. A veces, la pista de un fallo se encuentra dentro de algunas salidas básicas que puede recopilar del dispositivo. En este documento, tiene ejemplos de cómo aprovechar Python Scripts para obtener estos datos. Se consideran tres protocolos: OSPF, EIGRP e IS-IS.

Paso 1. Lo primero que debe hacer es configurar y habilitar guest shell.

```
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
```

En esta configuración, hay tres pasos importantes:

1. Active el servicio IOX. Esto es necesario para habilitar el shell de invitado.

2. Configure el VirtualPortGroup que actúa como el gateway predeterminado para el gateway predeterminado de guestShell.

3. Configure el alojamiento de aplicaciones para el shell de invitado. Las configuraciones le

permiten saber dónde entra en juego VirtualPortGroup.

Paso 2. A continuación, debe habilitar guest shell desde el modo de privilegio.

```
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 todo está correctamente configurado, debe ver el registro en el ejemplo anterior.

Paso 3. Ahora está listo para configurar los scripts python. Ejecute el comando guestshell en modo de privilegio. Verá el mensaje como en el siguiente ejemplo:

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

Paso 4. Cree un archivo con vi editor y configure los scripts basados en los protocolos que ha habilitado.

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

Esta ventana aparece

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

Paso 5. Presione "i" para insertar texto. Pegue la secuencia de comandos, pulse "esc" y, a continuación, introduzca los caracteres :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 ~]$
```

Salga del shell invitado con el comando exit.

Verificación

Pruebe el script. Salga del shell invitado con el comando exit. Luego ejecute guest shell run python3 ospf.py

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

Estos son los scripts para los tres protocolos; OSPF, EIGRP e 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")
```

Puede automatizar la recopilación de registros con scripts EEM que ejecutan los scripts Python después de observar los patrones de syslog. En la siguiente sección, tiene los scripts EEM que puede configurar junto con los scripts python para realizar esta tarea.

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
```

Nota: Los comandos recopilados en este script proporcionan información inicial básica. Al abrir un caso de TAC, los ingenieros del TAC pueden solicitar más información para investigar más a fondo si es necesario.

Enlaces de referencia

- Concha de invitados
- API de Python

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).