Coletar logs de não-sincronização do protocolo de roteamento IOS-XE com Python

Contents

Introdução

Este documento descreve como configurar scripts Python para coletar logs OSPF, EIGRP e IS-IS quando os protocolos não sincronizam.

Pré-requisitos

Requisitos

A Cisco recomenda que você esteja familiarizado com os tópicos listados:

- Configuração de hospedagem de aplicativo
- OSPF
- EIGRP
- IS-IS
- editor vi

Componentes Utilizados

As informações neste documento são baseadas no software Cisco IOS XE versão 17.

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Note: Este documento não se aprofunda nos detalhes do apphosting. Mais informações podem ser encontradas nos links mencionados.

Configurar

Configurações

Ao abrir um caso de TAC, é muito importante coletar informações relevantes para economizar tempo. Às vezes, a pista para uma falha está dentro de algumas saídas básicas que você pode coletar do dispositivo. Neste documento, você tem exemplos de como utilizar scripts Python para obter esses dados. Três protocolos são considerados: OSPF, EIGRP e IS-IS.

Etapa 1. A primeira coisa que você precisa fazer é configurar e ativar o 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
```

Nessa configuração, há três etapas importantes:

1. Ative o serviço IOX. Isso é necessário para habilitar o guestshell.

2. Configure o VirtualPortGroup que atua como o gateway padrão para o gateway padrão de convidado.

3. Configure a hospedagem de aplicativos para o convidado. Você pode saber pelas

configurações onde o VirtualPortGroup entra em ação.

Etapa 2. Em seguida, você precisa ativar o guestshell no modo privilegiado.

```
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
```

Se tudo estiver configurado corretamente, você deverá ver o log no exemplo anterior.

Etapa 3. Agora, você está pronto para configurar os scripts python. Execute o comando guestshell no modo privilegiado. Você verá o prompt como no exemplo subsequente:

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

Etapa 4. Crie um arquivo com o editor vi e configure os scripts com base nos protocolos ativados.

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

Essa janela é exibida

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

Etapa 5. Pressione "i" para inserir o texto. Cole o script e pressione "esc" e insira os 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 ~]$
```

Saia do guestshell com o comando exit.

Verificar

Teste o script. Saia do guestshell com o comando exit. Em seguida, execute guestshell run python3 ospf.py

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

Aqui estão os scripts de todos os três 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")
```

Você pode automatizar a coleta de logs com scripts EEM que executam os scripts Python depois que os padrões de syslog são observados. Na próxima seção, você tem os scripts EEM que pode configurar junto com os scripts python para realizar essa tarefa.

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
```

Note: Os comandos coletados nesse script fornecem informações iniciais básicas. Quando você abre um caso de TAC, engenheiros do TAC podem solicitar mais informações para investigar mais, se necessário.

Links de referência

- Concha de convidado
- API Python

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.