أتمتة محولات Catalyst 9000 باستخدام برامج Python النصية

المحتويات

- المقدمة
- المتطلبات الأساسية
  - المتطلبات
  - المكونات المستخدمة
  - الاصطلاحات
- معلومات أساسية
  - نصوص Guest Shell و Python النصية
  - تواجد استخدام بايثون مع مخطوطات معدد الداخل والخارج
  - اعتبارات استخدام بايثون مع مخطوطات IM
  - Sellnux في Cisco IOS XE
- التكوين
  - تمكين Guest Shell مع اكتساب عنوان IP ديناميكي
  - تمكين طبقة الضيف باستخدام عنوان DHCP IP
- حالات الاستخدام
  - استخدام الحالة 1: الحفاظ التلقائي لتغييرات التكوين على خادم SCP
  - استخدم الحالة 2: مراقبة زيادات في تغييرات مخطط STP
- معلومات ذات صلة

المقدمة

يصف هذه الوثيقة كيف أن يمكن IM مع Python برمجية لأتمتة تشكيل وجمع معطيات على مادة حفازة 9000 مفتاح.

المتطلبات الأساسية

المتطلبات

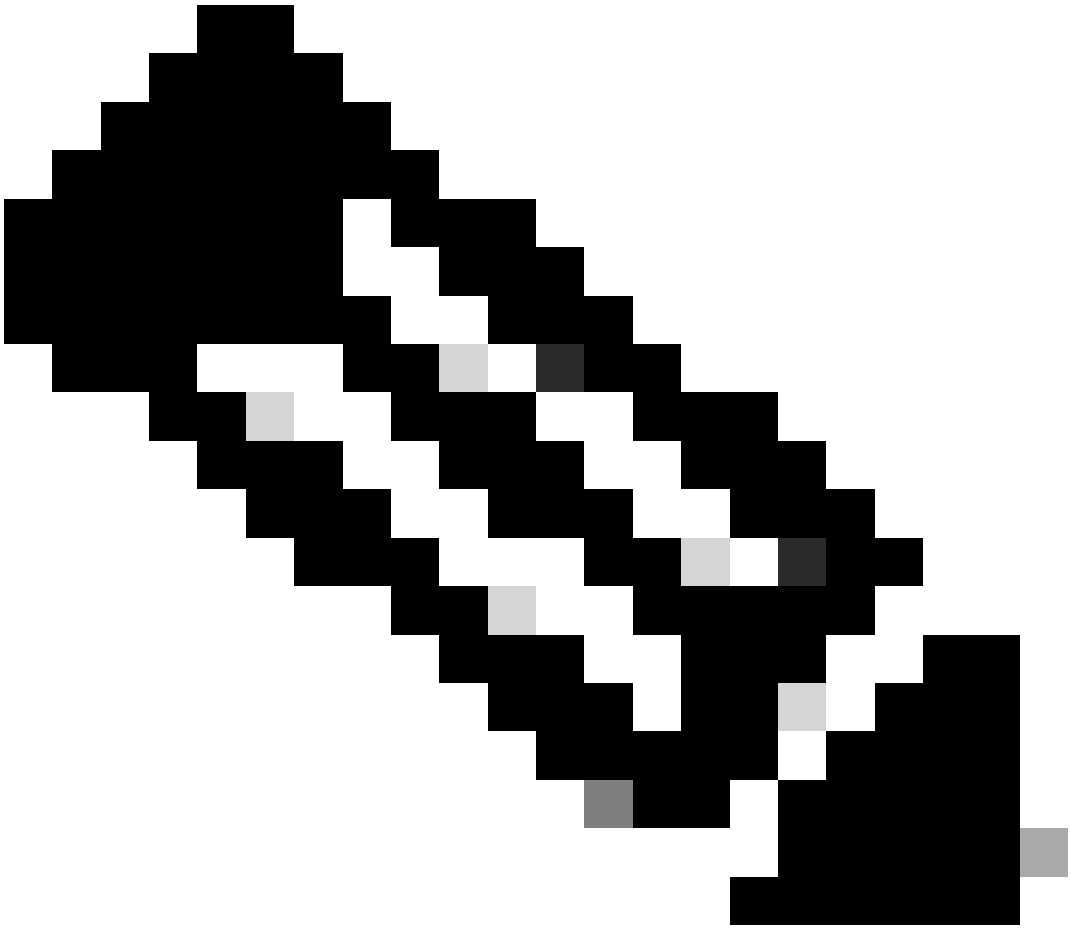
توصي Cisco بأن تكون لديك معرفة بالمواضيع التالية ودراية بها:

- IOS® و Cisco IOS® XE IM
- إستضافة التطبيقات و Guest Shell
- برمجة بايثون
- أوامر Linux

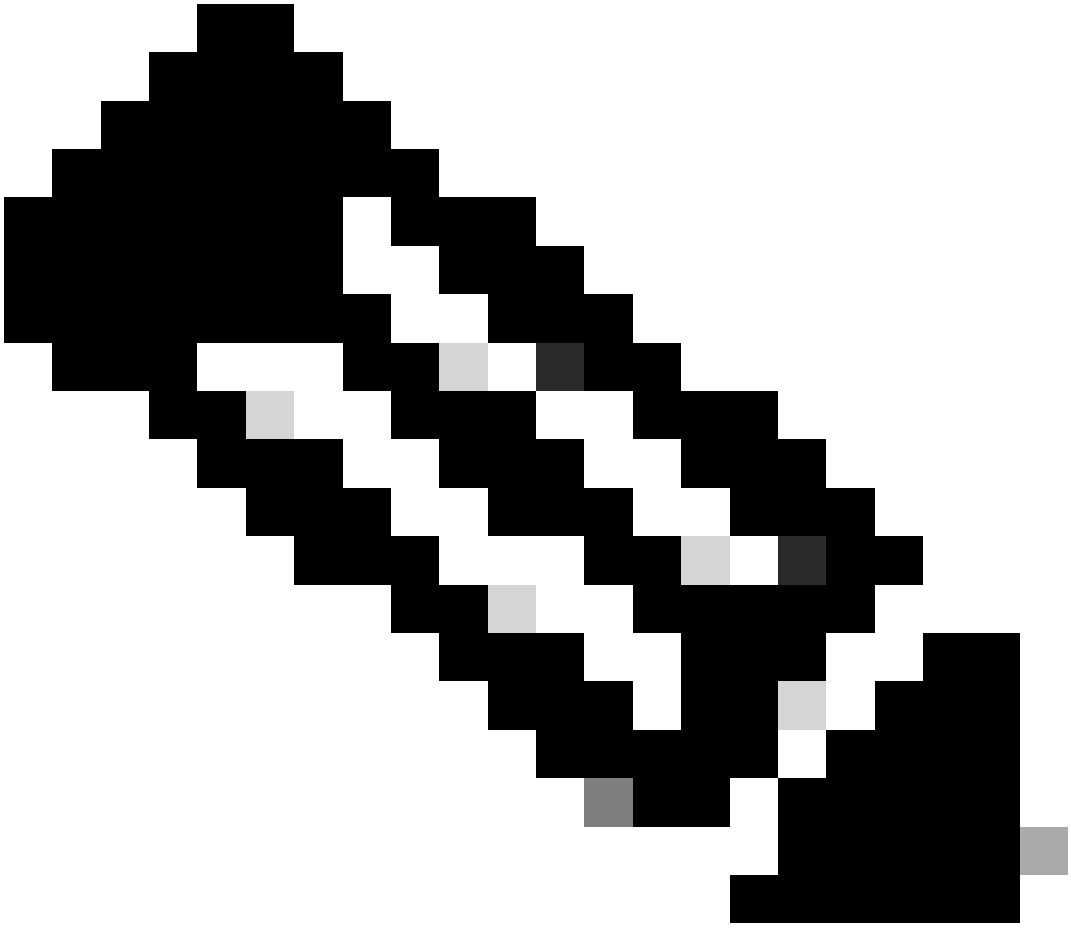
المكونات المستخدمة

تستند المعلومات الواردة في هذا المستند إلى إصدارات البرامج والمكونات المادية التالية:

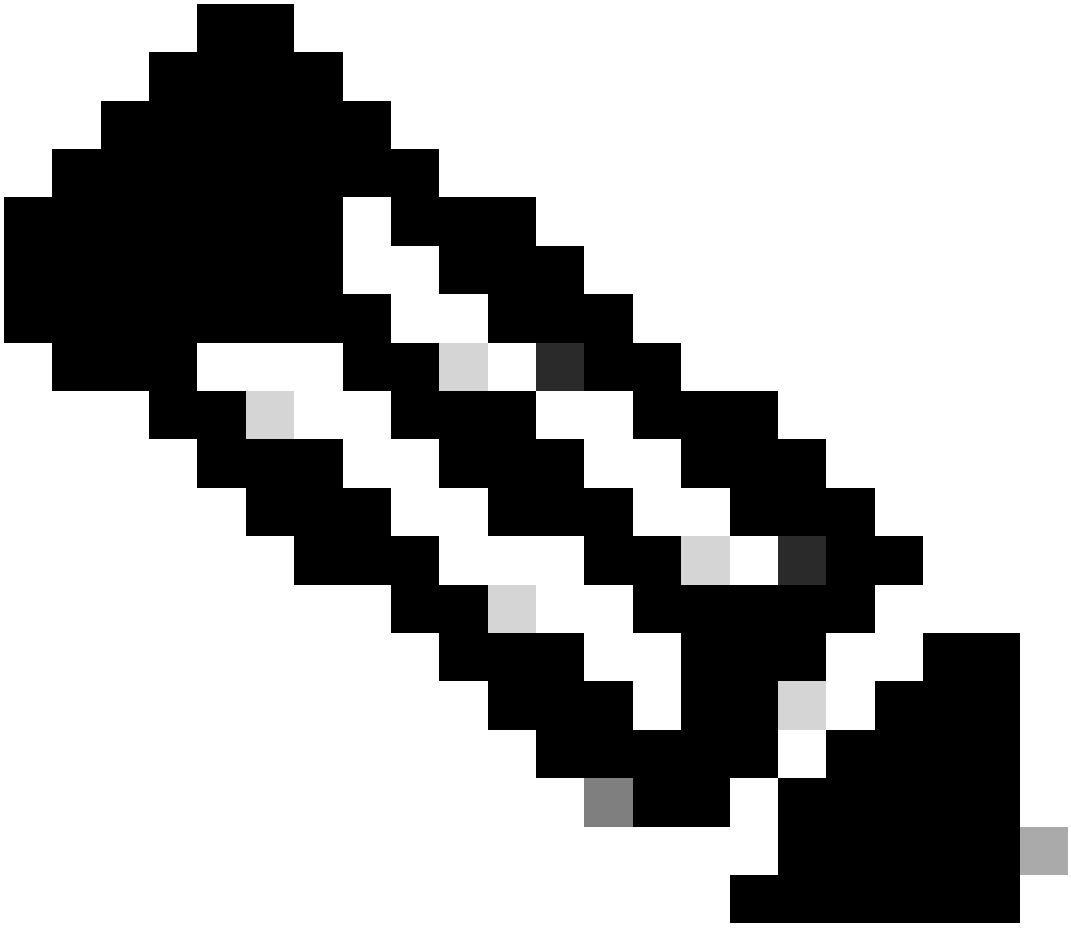
- Catalyst 9200
- Catalyst 9300
- Catalyst 9400
- Catalyst 9500
- Catalyst 9600
- Cisco IOS XE 17.9.1 والإصدارات الأحدث



ملاحظة: راجع دليل التكوين المناسب لمعرفة الأوامر المستخدمة لتمكين هذه الميزات على أنظمة Cisco الأساسية الأخرى.



ملاحظة: لا يساند مادة حفازة 9200L مفتاح Guest Shell.



ملاحظة: لا تدعم Cisco TAC هذه البرامج النصية ويتم توفيرها على أساس ما هو متوفر لأغراض تعليمية.

تم إنشاء المعلومات الواردة في هذا المستند من الأجهزة الموجودة في بيئة معملية خاصة. بدأت جميع الأجهزة المُستخدمة في هذا المستند بتكوين ممسوح (افتراضي). إذا كانت شبكتك قيد التشغيل، فتأكد من فهمك للتأثير المحتمل لأي أمر.

الاصطلاحات

أحلت Cisco فني طرف إتفاق لمعلومة على وثيقة إتفاق.

معلومات أساسية

نصوص Guest Shell و Python النصية

توفر إستضافة التطبيقات على مجموعة المحولات Cisco Catalyst 9000 فرص الابتكار

للشركاء والمطورين، حيث يمكن دمج أجهزة الشبكة مع بيئة وقت تشغيل التطبيق.

وهو دعم التطبيقات المحتواة، مما يوفر عزلة كاملة عن نظام التشغيل الرئيسي وعنصر kernel لبرنامج Cisco IOS XE. يضمن هذا الفصل تمييز عمليات تخصيص الموارد للتطبيقات المستضافة عن وظائف التوجيه والتحويل الأساسية.

تعرف البنية الأساسية لاستضافة التطبيق لأجهزة Cisco IOS XE باسم IOx (Cisco IOS + Linux)، والذي يسهل إستضافة التطبيقات والخدمات التي تم تطويرها بواسطة Cisco والشركاء والمطورين التابعين لجهات خارجية على أجهزة الشبكة، مما يضمن التكامل السلس عبر منصات الأجهزة المتنوعة.

تمثل Guest Shell، وهي عملية نشر متخصصة للحاويات، نموذجا لتطبيق مفيد لنشر الأنظمة.

توفر Guest Shell بيئة افتراضية قائمة على بيئة Linux تم تصميمها لتشغيل تطبيقات Linux المخصصة، بما في ذلك Python، لتمكين التحكم والإدارة المؤتمتين لأجهزة Cisco. يسمح حاوية Guest Shell للمستخدمين بتشغيل البرامج النصية والتطبيقات داخل النظام. على وجه التحديد، في منصات عمل Intel X86، تكون حاوية Guest Shell حاوية Linux (LXC) مع نظام ملفات جذري الحد الأدنى لنظام التشغيل CentOS 8. 0. في الإصدار 17.3.1 من Cisco IOS XE Amsterdam والإصدارات الأحدث، يتم دعم الإصدار 3.6 فقط من Python. يمكن تركيب مكتبات Python الإضافية للتشغيل باستخدام الأداة المساعدة Yum في CentOS 8.0. كما يمكن تثبيت حزم Python أو تحديثها باستخدام حزم PIP (PIP).

تتضمن Guest Shell واجهة برمجة تطبيقات Python (API)، والتي تسمح بتشغيل أوامر Cisco IOS XE باستخدام وحدة واجهة سطر الأوامر (CLI) لـ Python. بهذه الطريقة، تعزز برامج Python النصية الإمكانيات الأتمتة، وتوفر لمهندسي الشبكات أداة متعددة الاستخدامات لتطوير برامج نصية لأتمتة مهام التكوين وجمع البيانات. بينما يمكن تشغيل هذه البرامج النصية يدويا من خلال واجهة سطر الأوامر، يمكن أيضا إستخدامها بجانب برامج IM النصية للاستجابة لأحداث معينة، مثل رسائل syslog أو أحداث الواجهة أو تشغيل الأوامر. ومن الناحية العملية، يمكن إستخدام أي حدث يمكن أن يشغل برنامج نصي IM أيضا لتشغيل برنامج نصي لـ Python، مما يعمل على توسيع إمكانات الأتمتة داخل محولات Cisco Catalyst 9000 switches.

عند تثبيت Guest Shell، يتم إنشاء دليل مشاركة الضيف تلقائيا في نظام ملف Flash. Guest Shell و Python لها حق الوصول إليه من البرامج النصية. هذا هو نظام الملفات الذي يمكن الوصول إليه من خارج Guest Shell. هذا المجلد أقل من 50 ميغابايت. تأكد من مراعاة المساحة المتاحة عند إستخدام التجميع، لضمان الأداء السليم.

فوائد إستخدام بايثون مع مخطوطات معدة الإدخال والإخراج

- يوفر Python بتوسيع إمكانات التشغيل التلقائي لسجلات IM من خلال السماح بمعالجة المنطق المعقد (مثل التعبيرات العادية، حلقات التكرار، والمطابقات) ضمن البرنامج النصي Python. توفر هذه الإمكانية فرصة لإنشاء نصوص وظيفية أكثر قوة.

- ولا يتطلب تعلم لغة برمجة معروفة، تعمل Python على خفض الحواجز التي تحول دون دخول وبالإضافة إلى ذلك، مهندسي الشبكة الذين يرغبون في أتمتة أجهزة Cisco IOS XE. توفر هذه التقنية الإمكانية المحافظة والقراءة.

- كما توفر Python إمكانات معالجة الأخطاء، فضلا عن مكتبة قياسية قوية.

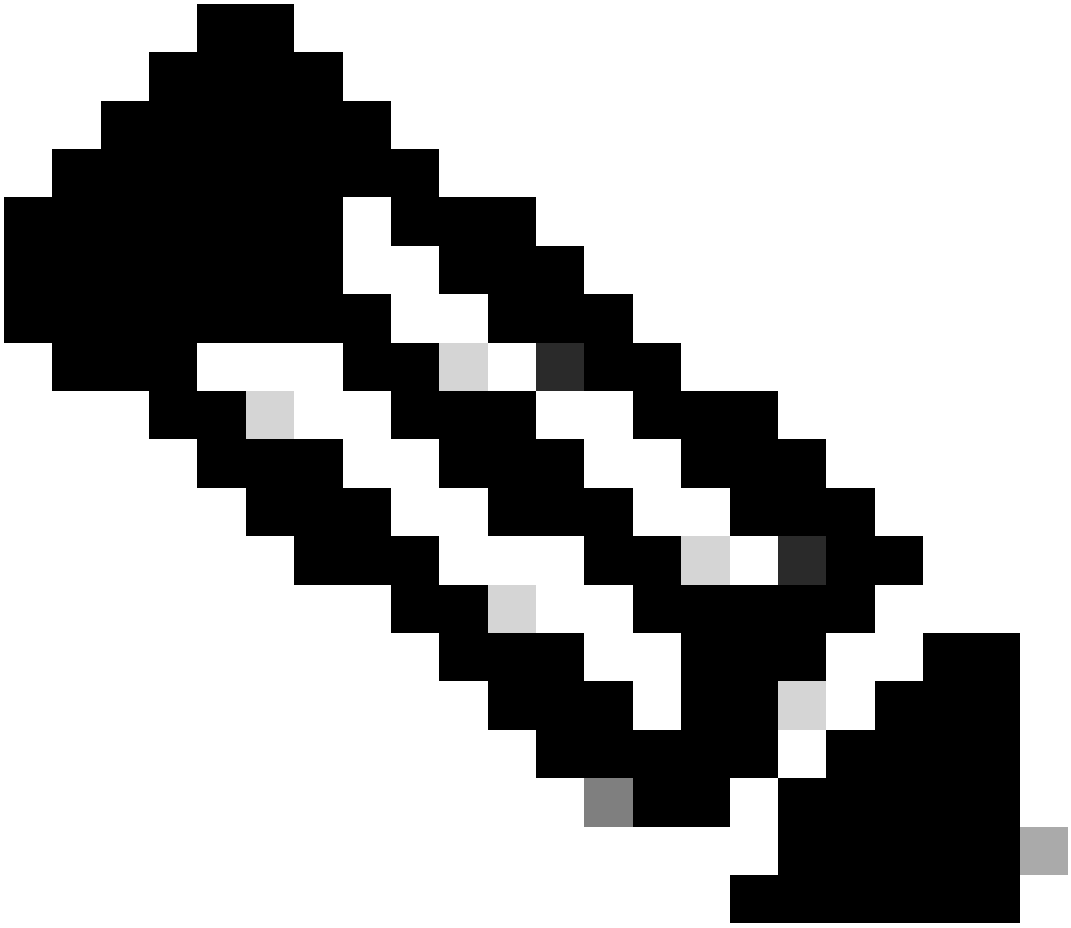
اعتبارات إستخدام بايثون مع مخطوطات IM

- يتم تمكين Guest Shell بشكل افتراضي، لذلك يجب تمكينها قبل تشغيل برامج Python النصية.
- يمكن إنشاء نصوص Python النصية مباشرة في CLI؛ هم يحتاجون أن يكونوا مخزنة في بيئة تطوير وبعد ذلك نسخت إلى ذاكرة فلاش من المفتاح.

SeLinux في Cisco IOS XE

بدءا من Cisco IOS XE 17.8.1، تم تقديم الدعم لنظام التشغيل Linux المحسن للأمان (SELinux) لفرض الأمان من خلال السياسات التي تحكم كيفية تفاعل العمليات والمستخدمين والملفات مع بعضها البعض. تحدد سياسة SELinux الإجراءات والموارد المسموح بالوصول إليها بواسطة عملية أو مستخدم. قد يحدث انتهاك عندما يحاول مستخدم أو عملية تنفيذ إجراء غير مسموح به من قبل السياسة، على سبيل المثال، الوصول إلى مورد أو تشغيل أمر. يمكن أن تعمل SeLinux في وضعين مختلفين:

1. وضع السماح: الموقت SeLinux بفرض أي سياسات. غير أنها لا تزال تسجل أي انتهاكات كما لو كانت مرفوضة.
2. فرض: تعمل SeLinux بشكل نشط على فرض سياسات الأمان على النظام. إذا خالف إجراء سياسة SELinux، يتم رفض الإجراء وتسجيله.



ملاحظة: تم تعيين الوضع الافتراضي على السماح عند تقديمه في Cisco IOS XE 17.14.1، يتم تمكين SELinux في وضع الإنفاذ. ومع ذلك، بدءا من الإصدار 17.8.1.

عند إستخدام Guest Shell، يمكن رفض الوصول إلى بعض الموارد عند إستخدام وضع التنفيذ. إذا نتج عن محاولة تنفيذ إجراء باستخدام Guest Shell أو برنامج نصي Python خطأ رفض الإذن، مماثل لهذا السجل:

```
*Jan 21 13:22:01: %SELINUX-1-VIOLATION: Chassis 1 R0/0: audispd: type=AVC msg=audit(1738074795.448:198)
```

للتحقق من رفض البرنامج النصي من قبل SELinux، أستخدم الأمرshow platform software audit summary للتحقق مما إذا كان عدد مرات الرفض يتزايد. وبالإضافة إلى ذلك،show platform software audit allيعرض سجل بالإجراءات التي تم حظرها بواسطة SELinux. ذاكرة التخزين المؤقت لمتجه الوصول (AVC) هي الآلية المستخدمة لتسجيل قرارات التحكم في الوصول في SELinux، لذلك عند إستخدام هذا الأمر، ابحث عن السجلات التي تبدأ بالنوع=AVC.

إذا تم رفض برنامج نصي وتم حظره، يمكن تعيين SELinux على وضع السماح باستخدام الأمرset platform software selinux permissive. وأيضًا تشغيله في التكوين الجاري في التغيير الذي تم تخزينه في التكوين. لذلك، كل مرة المفاتيح يعد التشغيل، لذلك بعد إعادة التحميل، يعود الوضع إلى التنفيذ. لذلك، قبل تغيير أي شيء، تأكد من التحقق من التغيير باستخدامshow platform software selinux.

التكوين

لأتمتة المهام على المحول باستخدام برامج IM و Python النصية، اتبع الخطوات التالية:

1. قم بتمكين Guest Shell.
2. انسخ البرنامج النصي Python إلى /flash/guest-share/ directory. يمكنك إستخدام أي آلية نسخ متوفرة في Cisco IOS XE، مثل SCP أو FTP أو مدير Flle في WebUI. يجب أن يكون البرنامج النصي Python في ذاكرة Flash، يمكنك تشغيله باستخدام الأمرguestshell run python3 /flash/guest-share/cat9k_script.py.
3. قم بتكوين برنامج نصي ل IM يشغل برنامج Python النصي. يسمح هذا الإعداد بتشغيل برنامج Python النصي باستخدام أي من كاشفات الأحداث المتعددة التي توفرها برامج IM النصية، مثل رسالة syslog، ومطابقة CLI، ووجود Cron.

في هذا القسم، يتم شرح الخطوة 1. يقدم القسم التالي أمثلة توضح كيفية تنفيذ الخطوات 2 و 3.

تمكين Guest Shell مع عنوان إستاتيكي

اتبع هذه العملية لتمكين Guest Shell:

1. تمكين IOx.

```
<#root>

Switch#conf t
Switch(config)#iox
Switch(config)#
*Feb 17 18:13:24.440: %UICFGEXP-6-SERVER_NOTIFIED_START: Switch 1 R0/0: psd: Server iox has been r
*Feb 17 18:13:28.797: %IOX-3-IOX_RESTARTABITLITY: Switch 1 R0/0: run_ioxn_caf: Stack is in N+1 mod

*Feb 17 18:13:36.069: %IM-6-IOX_ENABLEMENT: Switch 1 R0/0: ioxman: IOX is ready.
```

2. قم بتكوين شبكة إستضافة التطبيقات ل Guest Shell. يستخدم هذا المثال واجهة AppGigabitEthernet لتوفير الوصول إلى الشبكة؛ وعموما، يمكنك أيضا إستخدام واجهة الإدارة (Gi0/0).

```
Switch(config)#int appgig1/0/1
Switch(config-if)#switchport mode trunk
Switch(config-if)#switchport trunk allowed vlan 20
```

```
Switch(config)#app-hosting appid guestshell
Switch(config-app-hosting)#app-vnic appGigabitEthernet trunk
Switch(config-config-app-hosting-trunk)#vlan 20 guest-interface 0
Switch(config-config-app-hosting-vlan-access-ip)#guest-ipaddress 10.20.1.2 netmask 255.255.255.0
Switch(config-config-app-hosting-vlan-access-ip)#exit
Switch(config-config-app-hosting-trunk)#exit
Switch(config-app-hosting)#app-default-gateway 10.20.1.1 guest-interface 0
Switch(config-app-hosting)#name-server0 10.31.104.74
Switch(config-app-hosting)#end
```

3. قم بتمكين Guest Shell.

```
<#root>

Switch#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
```

4. تحقق من صحة Guest Shell. يتحقق هذا المثال من توفر إمكانية الوصول باستخدام البوابة الافتراضية وكذلك مع cisco.com. أيضا، تأكد من أن Python 3 يستطيع كنت شغلت من الصيف طبقة.

```
<#root>

! Validate that the Guest Shell is running.
Switch#

show app-hosting list


App id                                   State
------------------------------------------------------------

guestshell


RUNNING


Switch#

guestshell run bash

[guestshell@guestshell ~]$

! Validate that the IP address of the Guest Shell is configured correctly.
```

```
[guestshell@guestshell ~]$
```

sudo ifconfig

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

inet 10.20.1.2 netmask 255.255.255.0

```
  broadcast 10.20.1.255
        inet6 fe80::5054:ddff:fe61:24c7  prefixlen 64  scopeid 0x20
        ether 52:54:dd:61:24:c7  txqueuelen 1000  (Ethernet)
        RX packets 23  bytes 1524 (1.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 9  bytes 726 (726.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 177  bytes 34754 (33.9 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 177  bytes 34754 (33.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

! Validate reachability to the default gateway and ensure that DNS is resolving correctly.
[guestshell@guestshell ~]$
```

ping 10.20.1.1

```
PING 10.20.1.1 (10.20.1.1) 56(84) bytes of data.
64 bytes from 10.20.1.1: icmp_seq=2 ttl=254 time=0.537 ms
64 bytes from 10.20.1.1: icmp_seq=3 ttl=254 time=0.537 ms
64 bytes from 10.20.1.1: icmp_seq=4 ttl=254 time=0.532 ms
64 bytes from 10.20.1.1: icmp_seq=5 ttl=254 time=0.574 ms
64 bytes from 10.20.1.1: icmp_seq=6 ttl=254 time=0.590 ms
^C
--- 10.20.1.1 ping statistics ---
6 packets transmitted, 5 received, 16.6667% packet loss, time 5129ms
rtt min/avg/max/mdev = 0.532/0.554/0.590/0.023 ms

[guestshell@guestshell ~]$
```

ping cisco.com

```
PING cisco.com (X.X.X.X) 56(84) bytes of data.
64 bytes from www1.cisco.com (X.X.X.X): icmp_seq=1 ttl=237 time=125 ms
64 bytes from www1.cisco.com (X.X.X.X): icmp_seq=2 ttl=237 time=125 ms
^C
--- cisco.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 124.937/125.141/125.345/0.204 ms

! Validate the Python version.
[guestshell@guestshell ~]$
```

python3 --version

Python 3.6.8

```
! Run Python commands within the Guest Shell.
[guestshell@guestshell ~]$
```

```
python3

Python 3.6.8 (default, Dec 22 2020, 19:04:08)
[GCC 8.4.1 20200928 (Red Hat 8.4.1-1)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>

print("Cisco")
Cisco

>>> exit()
[guestshell@guestshell ~]$

[guestshell@guestshell ~]$ exit
exit

Switch#
```

تمكين طبقة الضيف باستخدام عنوان DHCP IP

عادة، يتم تكوين Guest Shell باستخدام عنوان IP ثابت لأن حاوية Guest Shell لا تحتوي على خدمة عميل DHCP بشكل افتراضي. إذا كان من الضروري أن يطلب Guest Shell عنوان IP بشكل ديناميكي، يلزم تثبيت خدمة عميل DHCP. اتبع هذه العملية:

1. اتبع الخطوات لتمكين Guest Shell باستخدام عنوان IP ثابت. ومع ذلك، لا تقم هذه المرة بتعيين عنوان IP في تكوين إستضافة التطبيقات أثناء الخطوة 2. وبدلا من ذلك، أستخدم هذا التكوين:

```
Switch(config)#int appgig1/0/1
Switch(config-if)#switchport mode trunk
Switch(config-if)#switchport trunk allowed vlan 20

Switch(config)#app-hosting appid guestshell
Switch(config-app-hosting)#app-vnic appGigabitEthernet trunk
Switch(config-config-app-hosting-trunk)#vlan 20 guest-interface 0
Switch(config-app-hosting)#end
```

2. يمكن تثبيت عميل DHCP باستخدام الأداة المساعدة Yum باستخدام الأمرsudo yum install dhcp-client. ومع ذلك، تم إلغاء صلاحية مستودعات تدفق CentOS 8. لمعالجة هذه المشكلة، يمكن تنزيل حزم عميل DHCP وتثبيتها يدويا. على جهاز الكمبيوتر، قم بتنزيل هذه الحزم من مساحة التخزين CentOS Stream 8 أو حزمها في ملف tar.

   - bind-export-libs-9.11.36-13.el8.x86_64.rpm
   - dhcp-client-4.3.6-50.el8.x86_64.rpm
   - dhcp-common-4.3.6-50.el8.noarch.rpm
   - dhcp-libs-4.3.6-50.el8.x86_64.rpm

```
[cisco@CISCO-PC guestshell-packages] % tar -cf dhcp-client.tar bind-export-libs-9.11.36-13.el8.x86
```

3. انسخ الملفdhcp-client.tarإلى /flash/guest-share/directory في الموجل.

4. أدخل جلسة عمل Guest Shell وقم بتشغيل أوامر Linux لتثبيت عميل DHCP وطلب عنوان IP.

```
<#root>

513E.D.02-C9300X-12Y-A-17#

guestshell run bash


[guestshell@guestshell ~]$

sudo ifconfig

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536

<--- no eth0 interface

        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 149  bytes 32462 (31.7 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 149  bytes 32462 (31.7 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

! Unpack the packages needed for the DHCP client service.
[guestshell@guestshell ~]$

tar -xf /flash/guest-share/dhcp-client.tar

tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.metadata:kMDItemWhereFro
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.macl'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.metadata:kMDItemWhereFro
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.macl'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.metadata:kMDItemWhereFro
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.macl'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.quarantine'
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.metadata:kMDItemWhereFro
tar: Ignoring unknown extended header keyword 'LIBARCHIVE.xattr.com.apple.macl'

[guestshell@guestshell ~]$

ls

bind-export-libs-9.11.36-13.el8.x86_64.rpm  dhcp-common-4.3.6-50.el8.noarch.rpm
dhcp-client-4.3.6-50.el8.x86_64.rpm         dhcp-libs-4.3.6-50.el8.x86_64.rpm

! Install the packages using DNF.
[guestshell@guestshell ~]$

sudo dnf -y --disablerepo=* localinstall *.rpm

Warning: failed loading '/etc/yum.repos.d/CentOS-Base.repo', skipping.
Dependencies resolved.
```

```
================================================================================================
 Package                    Architecture    Version                       Repository            Siz
================================================================================================
Installing:
 bind-export-libs           x86_64          32:9.11.36-13.el8             @commandline          1.1
 dhcp-client                x86_64          12:4.3.6-50.el8               @commandline          319
 dhcp-common                noarch          12:4.3.6-50.el8               @commandline          208
 dhcp-libs                  x86_64          12:4.3.6-50.el8               @commandline          148

Transaction Summary
================================================================================================
Install  4 Packages

Total size: 1.8 M
Installed size: 3.9 M
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :                                                                        1/
  Installing       : dhcp-libs-12:4.3.6-50.el8.x86_64                                       1/
  Installing       : dhcp-common-12:4.3.6-50.el8.noarch                                     2/
  Installing       : bind-export-libs-32:9.11.36-13.el8.x86_64                              3/
  Running scriptlet: bind-export-libs-32:9.11.36-13.el8.x86_64                              3/
  Installing       : dhcp-client-12:4.3.6-50.el8.x86_64                                     4/
  Running scriptlet: dhcp-client-12:4.3.6-50.el8.x86_64                                     4/
  Verifying        : bind-export-libs-32:9.11.36-13.el8.x86_64                              1/
  Verifying        : dhcp-client-12:4.3.6-50.el8.x86_64                                     2/
  Verifying        : dhcp-common-12:4.3.6-50.el8.noarch                                     3/
  Verifying        : dhcp-libs-12:4.3.6-50.el8.x86_64                                       4/

Installed:
  bind-export-libs-32:9.11.36-13.el8.x86_64           dhcp-client-12:4.3.6-50.el8.x86_64
  dhcp-common-12:4.3.6-50.el8.noarch                  dhcp-libs-12:4.3.6-50.el8.x86_64

Complete!

! Request a DHCP IP address for eth0.
[guestshell@guestshell ~]$
```

sudo dhclient eth0

```
! Validate the leased IP address.
[guestshell@guestshell ~]$
```

sudo ifconfig eth0

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

inet 10.1.1.12 netmask 255.255.255.0

```
  broadcast 10.1.1.255
        inet6 fe80::5054:ddff:fe85:a0d5  prefixlen 64  scopeid 0x20
        ether 52:54:dd:85:a0:d5  txqueuelen 1000  (Ethernet)
        RX packets 7  bytes 1000 (1000.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11  bytes 1354 (1.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
[guestshell@guestshell ~]$

exit

exit

! You can validate the leased IP address from Cisco IOS XE too.
513E.D.02-C9300X-12Y-A-17#

guestshell run sudo ifconfig eth0

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500


inet 10.1.1.12  netmask 255.255.255.0

  broadcast 10.1.1.255
        inet6 fe80::5054:ddff:fe85:a0d5  prefixlen 64  scopeid 0x20
        ether 52:54:dd:85:a0:d5  txqueuelen 1000  (Ethernet)
        RX packets 28  bytes 2344 (2.2 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 13  bytes 1494 (1.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

حالات الاستخدام

إستخدام الحالة 1: الحفظ التلقائي لتغييرات التكوين على خادم SCP

في بعض الحالات، من المفيد أن تقوم تلقائيا بحفظ تكوينات المحول إلى خادم في كل مرة يتم write memory إستخدام الأمر. وتساعد هذه الممارسة على الاحتفاظ بسجل للتغييرات وتسمح بالعودة إلى حالة التكوين السابقة إذا لزم الأمر. عند إختيار خادم، يمكن إستخدام كل من TFTP و SCP، ولكن خادم SCP يوفر طبقة إضافية من الأمان.

توفر ميزة أرشيف Cisco IOS هذه الوظيفة. ومع ذلك، يتمثل أحد العيوب المهمة في أنه ال يمكن إخفاء بيانات اعتماد بروتوكول SCP في التكوين؛ يتم عرض مسار الخادم في نص عادي في كل من تكوينات التشغيل وبدء التشغيل.

```
Switch#show running-config | section archive
archive
 path scp://cisco:Cisco!123@10.31.121.224/
 write-memory
```

من خلال إستخدام Guest Shell و Python، من الممكن تحقيق نفس الوظيفة مع إخفاء بيانات الخادم في متغيرات البيئة داخل Guest Shell. ويتم تحقيق ذلك من خلال الاستفادة من نفس الوقت. ونتيجة لذلك، ال تكون بيانات اعتماد خادم SCP الفعلية. مرئية في التكوين الجاري تشغيله.

في هذه الطريقة، يعرض التكوين الجاري تشغيله فقط برنامج IM النصي، بينما يقوم copy

running-config scp:البرنامج النصي للبيثون بإنشاء الأمر باستخدام بيانات الاعتماد وتمريره إلى برنامج IM النصي ليتم تشغيله.

اتبع الخطوات التالية لهذا المثال:

1. انسخ البرنامج النصي Python إلى دليل /flash/guest-share. يقرأ هذا البرنامج النصي متغيرات البيئة SCP_USER و SCP_PASSWORD، ويعيد copy startup-config scp:الأمر، بحيث يمكن لبرنامج IM النصي تشغيله. يتطلب البرنامج النصي عنوان IP لخادم SCP وكوسيطة. بالإضافة إلى ذلك، فإن الأمر write memory يحافظ على سجل لكل مرة يتم فيها تشغيل الأمر في ملف مستمر موجود على /flash/guest-share/TAC-write-memory-log.txt. هذا هو نص بايثون:

```
import sys
import os
import cli
from datetime import datetime

# Get SCP server from the command-line argument (first argument passed)
scp_server = sys.argv[1]  # Expects the SCP server address as the first argument

# Configure CLI to suppress file prompts (quiet mode)
cli.configure("file prompt quiet")

# Get the current date and time
current_time = datetime.now()

# Format the current time for human-readable output and to use in filenames
formatted_time = current_time.strftime("%Y-%m-%d %H:%M:%S %Z")  # e.g., 2025-03-13 14:30:00 UTC
file_name_time = current_time.strftime("%Y-%m-%d_%H_%M_%S")  # e.g., 2025-03-13_14_30_00

# Retrieve SCP user and password from environment variables securely
scp_user = os.getenv('SCP_USER')  # SCP username from environment
scp_password = os.getenv('SCP_PASSWORD')  # SCP password from environment

# Ensure the credentials are set in the environment, raise error if missing
if not scp_user or not scp_password:
    raise ValueError("SCP user or password not found in environment variables!")

# Construct the SCP command to copy the file, avoiding exposure of sensitive data in print
# WARNING: The password should not be shared openly in logs or outputs.
print(f"copy startup-config scp://{scp_user}:{scp_password}@{scp_server}/config-backup-{file_name_

# Save the event in flash memory (log the write operation)
directory = '/flash/guest-share'  # Directory path where log will be saved
file_name = os.path.join(directory, 'TAC-write-memory-log.txt')  # Full path to log file

# Prepare the log entry with the formatted timestamp
line = f'{formatted_time}: Write memory operation.\n'

# Open the log file in append mode to add the new log entry
with open(file_name, 'a') as file:
    file.write(line)  # Append the log entry to the file
```

في هذا المثال، لا Python نسخت نص تنفيذي إلى المفتاح يستعمل TFTP لدان:

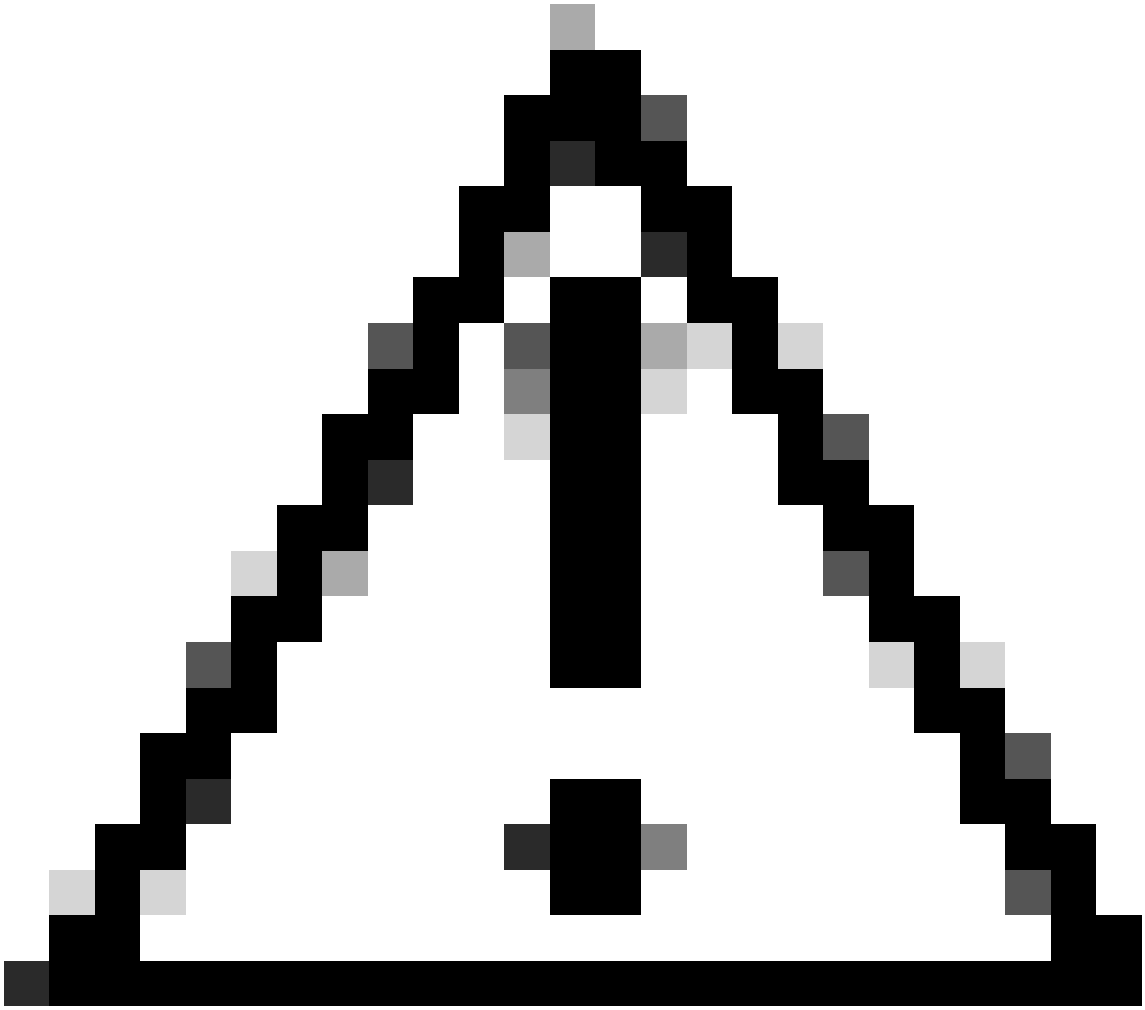
```
<#root>

Switch#

copy tftp://10.207.204.10/cat9k_scp_command.py flash:/guest-share/cat9k_scp_command.py


Accessing tftp://10.207.204.10/cat9k_scp_command.py...
Loading cat9k_scp_command.py from 10.207.204.10 (via GigabitEthernet0/0): !
[OK - 917 bytes]


917 bytes copied in 0.017 secs (53941 bytes/sec)
```

2. قم بتثبيت نص IM. هذا النص التنفيذي يدعو Python نصي. الذي يرجع copy startup-config scp: الأمر المطلوب لحفظ التكوين على خادم SCP. بعد ذلك يقوم برنامج IM النصي بتشغيل الأمر الذي تم إرجاعه من قبل برنامج Python النصي.

```
event manager applet Python-config-backup authorization bypass
 event cli pattern "^write|write memory|copy running-config startup-config" sync no skip no maxrur
 action 0000 syslog msg "Config save detected, TAC EEM-python started."
 action 0005 cli command "enable"
 action 0015 cli command "guestshell run python3 /bootflash/guest-share/cat9k_scp_command.py 10.31
 action 0020 regexp "(^.*)\n" "$_cli_result" match command
 action 0025 cli command "$command"
 action 0030 syslog msg "TAC EEM-python script finished with result: $_cli_result"
```

3. اضبط متغيرات بيئة Guest Shell بإدراجها في ~/.bashrc الملف. هذا يضمن أن متغيرات البيئة تكون ثابتة في كل مرة يتم فتح Guest Shell، حتى بعد إعادة تحميل المحول. إضافة هذين البندين:

```
export SCP_USER="cisco"
export SCP_PASSWORD="Cisco!123"
```



تحذير: وثائق الإعتماد المستخدمة في هذا المثال هي لأغراض تعليمية فقط. غير مخصصة للاستخدام في بيئات الإنتاج. يلزم على المستخدمين إستبدال بيانات الاعتماد هذه بوثائق الاعتماد الخاصة بهم الآمنة والخاصة بالبيئة.

هذه هي العملية لإضافة المتغيرات التالية إلى~/.bashrcالملف:

<#root>

! 1. Enter a Guest Shell bash session.
Switch#

guestshell run bash

! 2. Locate the ~/.bashrc file.
[guestshell@guestshell ~]$

ls ~/.bashrc

/home/guestshell/.bashrc

```
! 3. Add the SCP_USER and SCP_PASSWORD environment variables at the end of the ~/.bashrc file.
[guestshell@guestshell ~]$
echo 'export SCP_USER="cisco"' >> ~/.bashrc


[guestshell@guestshell ~]$
echo 'export SCP_PASSWORD="Cisco!123"' >> ~/.bashrc



! 4. To validate these 2 new lines were added correctly, display the content of the ~/.bashrc file
[guestshell@guestshell ~]$
cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific environment
if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]
then
    PATH="$HOME/.local/bin:$HOME/bin:$PATH"
fi
export PATH

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
[guestshell@guestshell ~]$ echo 'export SCP_USER="cisco"' >> ~/.bashrc
[guestshell@guestshell ~]$ echo 'export SCP_PASSWORD="Cisco!123"' >> ~/.bashrc
[guestshell@guestshell ~]$ cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific environment
if ! [[ "$PATH" =~ "$HOME/.local/bin:$HOME/bin:" ]]
then
    PATH="$HOME/.local/bin:$HOME/bin:$PATH"
fi
export PATH

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
export SCP_USER="cisco"
export SCP_PASSWORD="Cisco!123"

! 5. Reload the ~/.bashrc file in the current session.
[guestshell@guestshell ~]$
```

```
source ~/.bashrc


! 6. Validate that the environment variables are added, then exit the Guest Shell session.
[guestshell@guestshell ~]$

printenv | grep SCP
SCP_USER=cisco
SCP_PASSWORD=Cisco!123

[guestshell@guestshell ~]$ exit

Switch#
```

4. قم بتشغيل برنامج Python النصي ودوايا للتحقق من أنه يرجع الأمرcopyالأيمن وسجل العملية في الملف المستمرTAC-write-memory-log.txt.

```
<#root>

Switch#

guestshell run python3 /flash/guest-share/cat9k_scp_command.py 10.31.121.224
copy startup-config scp://cisco:Cisco!123@10.31.121.224/config-backup-2025-01-25_18_35_18.txt


Switch#

dir flash:guest-share/

Directory of flash:guest-share/

286725  -rw-              368  Jan 25 2025 18:35:18 +00:00

TAC-write-memory-log.txt

286726  -rw-              903  Jan 25 2025 18:34:45 +00:00  cat9k_scp_command.py
286723  -rw-              144  Jan 25 2025 15:07:07 +00:00  TAC-shutdown-log.txt
286722  -rw-              977  Jan 25 2025 14:50:56 +00:00  cat9k_noshut.py

11353194496 bytes total (3751542784 bytes free)

Switch#

more flash:/guest-share/TAC-write-memory-log.txt
2025-01-25 18:35:18 : Write memory operation.
```

5. اختبر نص IM. يرسل برنامج IM النصي هذا syslog أيضا مع نتيجة عملية النسخ، سواء كانت ناجحة أو فشلت. فيما يلي مثال على التشغيل الناجح:

```
<#root>

Switch#

write memory
```

```
Building configuration...
[OK]
Switch#
*Jan 25 19:23:22.189: %HA_EM-6-LOG: Python-config-backup: Config save detected, TAC EEM-python sta
*Jan 25 19:23:42.885: %HA_EM-6-LOG: Python-config-backup:

TAC EEM-python script finished with result:
Writing config-backup-2025-01-25_19_23_26.txt !
8746 bytes copied in 15.175 secs (576 bytes/sec)

Switch#


Switch#

more flash:guest-share/TAC-write-memory-log.txt
2025-01-25 19:23:26 : Write memory operation.
```

الاختبار عملية نقل فاشلة، يتم إيقاف تشغيل خادم SCP. هذه هي نتيجة هذه التشغيل الفاشل:

```
<#root>

Switch#

write

Building configuration...
[OK]
Switch#
*Jan 25 19:25:31.439: %HA_EM-6-LOG: Python-config-backup: Config save detected, TAC EEM-python sta
*Jan 25 19:26:06.934: %HA_EM-6-LOG: Python-config-backup:

TAC EEM-python script finished with result:
Writing config-backup-2025-01-25_19_25_36.txt % Connection timed out; remote host not responding

%Error writing scp://*:*@10.31.121.224/config-backup-2025-01-25_19_25_36.txt (Undefined error)

Switch#
Switch#

Switch#
Switch#

more flash:guest-share/TAC-write-memory-log.txt

2025-01-25 19:23:26 : Write memory operation.

2025-01-25 19:25:36 : Write memory operation.
```

أستخدم الحالة 2: مراقبة زيادات في تغييرات مخطط STP

هذا المثال مفيد لمراقبة المشاكل المتعلقة بعدم إستقرار الشجرة المتفرعة وتحديد الواجهة التي تتلقى إعلامات تغيير المخطط (TCNs). يتم تشغيل برنامج IM النصي بشكل دوري في فترة زمنية محددة ويستدعي برنامج نصي Python يشغل أمر show ويتحقق مما إذا كانت TCNs قد زادت.

إن إنشاء هذا النص التنفيذي باستخدام أوامر IM فقط يتطلب إستخدام حلقات التكرار ومطابقات regex المتعددة، والذي قد يكون مرهقا. لذلك، يوضح هذا المثال كيف أن نص IM Python لـ يفوض هذا المنطق المعقد.

اتبع الخطوات التالية لهذا المثال:

1. انسخ البرنامج النصي Python إلى /flash/guest-share/ directory. يقوم هذا البرنامج النصي بتنفيذ هذه المهام:

    1. هو يركض الأمرshow spanning-tree detailويحلل الإنتاج أن يحفظ TCN معلومة لكل VLAN في قاموس.

    2. إنه يقارن معلومات TCN التي تم تحليلها بالبيانات الموجودة في ملف JSON من تشغيل البرنامج النصي السابق. لكل شبكة محلية ظاهرية (VLAN)، إذا تمت زيادة شبكات TCN، يتم إرسال رسالة syslog بمعلومات مشابهة لهذا المثال:

```
*Jan 31 18:57:37.852: %GUESTSHELL-5-PYTHON_SCRIPT: Message from tty73(user id: shxUnknownTTY
```

    3. يحفظ معلومات TCN الحالية في ملف JSON ليتم مقارنتها أثناء التشغيل التالي. هذا هو نص بايثون:

```
import os
import json
import cli
import re
from datetime import datetime


def main():
    # Get TCNs by running the CLI command to show spanning tree details
    tcns = cli.cli("show spanning-tree detail")

    # Parse the output into a dictionary of VLAN details
    parsed_tcns = parse_stp_detail(tcns)

    # Path to the JSON file where VLAN TCN data will be stored
    file_path = '/flash/guest-share/tcns.json'

    # Initialize an empty dictionary to hold stored TCN data
    stored_tcn = {}

    # Check if the file exists and process it if it does
    if os.path.exists(file_path):
        try:
            # Open the JSON file and parse its contents into stored_tcn
            with open(file_path, 'r') as f:
                stored_tcn = json.load(f)
                result = compare_tcn_sets(stored_tcn, parsed_tcns)

                # Check each VLAN in the result and log changes if TCN increased
                for vlan_id, vlan_data in result.items():
```

```
                    if vlan_data['tcn_increased']:
                        log_message = (
                            f"TCNs increased in VLAN {vlan_id} "
                            f"from {vlan_data['old_tcn']} to {vlan_data['new_tcn']}. "
                            f"Last TCN seen on {vlan_data['source_interface']}."
                        )
                        # Send log message using CLI
                        cli.cli(f"send log facility GUESTSHELL severity 5 mnemonics PYTHON_S

            except json.JSONDecodeError:
                print("Error: The file contains invalid JSON.")
            except Exception as e:
                print(f"An error occurred: {e}")

        # Write the current TCN data to the JSON file for future comparison
        with open(file_path, 'w') as f:
            json.dump(parsed_tcns, f, indent=4)


    def parse_stp_detail(cli_output: str):
        """
        Parses the output of "show spanning-tree detail" into a dictionary of VLANs and their TC

        Args:
            cli_output (str): The raw output from the "show spanning-tree detail" command.

        Returns:
            dict: A dictionary where the keys are VLAN IDs and the values contain TCN details.
        """
        vlan_info = {}

        # Regular expressions to match various parts of the "show spanning-tree detail" output
        vlan_pattern = re.compile(r'^\s*(VLAN|MST)(\d+)\s*', re.MULTILINE)
        tcn_pattern = re.compile(r'^\s*Number of topology changes (\d+)\s*', re.MULTILINE)
        last_tcn_pattern = re.compile(r'last change occurred (\d+:\d+:\d+) ago\s*', re.MULTILINE
        last_tcn_days_pattern = re.compile(r'last change occurred (\d+d\d+h) ago\s*', re.MULTILI
        tcn_interface_pattern = re.compile(r'from ([a-zA-Z]+[\d+\/]+\d+)\s*', re.MULTILINE)

        # Find all VLAN blocks in the output
        vlan_blocks = vlan_pattern.split(cli_output)[1:]
        vlan_blocks = [item for item in vlan_blocks if item not in ["VLAN", "MST"]]

        for i in range(0, len(vlan_blocks), 2):
            vlan_id = vlan_blocks[i].strip()

            # Match the relevant patterns for TCN and related details
            tcn_match = tcn_pattern.search(vlan_blocks[i + 1])
            last_tcn_match = last_tcn_pattern.search(vlan_blocks[i + 1])
            last_tcn_days_match = last_tcn_days_pattern.search(vlan_blocks[i + 1])
            tcn_interface_match = tcn_interface_pattern.search(vlan_blocks[i + 1])

            # Parse the TCN details and add to the dictionary
            if last_tcn_match:
                tcn = int(tcn_match.group(1))
                last_tcn = last_tcn_match.group(1)
                source_interface = tcn_interface_match.group(1) if tcn_interface_match else None
                vlan_info[vlan_id] = {
                    "id_int": int(vlan_id),
                    "tcn": tcn,
                    "last_tcn": last_tcn,
                    "source_interface": source_interface,
                    "tcn_in_last_day": True
```

```
                }
            elif last_tcn_days_match:
                tcn = int(tcn_match.group(1))
                last_tcn = last_tcn_days_match.group(1)
                source_interface = tcn_interface_match.group(1) if tcn_interface_match else None
                vlan_info[vlan_id] = {
                    "id_int": int(vlan_id),
                    "tcn": tcn,
                    "last_tcn": last_tcn,
                    "source_interface": source_interface,
                    "tcn_in_last_day": False
                }

    return vlan_info


def compare_tcn_sets(set1, set2):
    """
    Compares two sets of VLAN TCN data to determine if TCN values have increased.

    Args:
        set1 (dict): The first set of VLAN TCN data.
        set2 (dict): The second set of VLAN TCN data.

    Returns:
        dict: A dictionary indicating whether the TCN has increased for each VLAN.
    """
    tcn_changes = {}

    # Compare TCN values for VLANs that exist in both sets
    for vlan_id, vlan_data_1 in set1.items():
        if vlan_id in set2:
            vlan_data_2 = set2[vlan_id]
            tcn_increased = vlan_data_2['tcn'] > vlan_data_1['tcn']
            tcn_changes[vlan_id] = {
                'tcn_increased': tcn_increased,
                'old_tcn': vlan_data_1['tcn'],
                'new_tcn': vlan_data_2['tcn'],
                'source_interface': vlan_data_2['source_interface']
            }
        else:
            tcn_changes[vlan_id] = {
                'tcn_increased': None,  # No comparison if VLAN is not in set2
                'old_tcn': vlan_data_1['tcn'],
                'new_tcn': None
            }

    # Check for VLANs in set2 that are not in set1
    for vlan_id, vlan_data_2 in set2.items():
        if vlan_id not in set1:
            tcn_changes[vlan_id] = {
                'tcn_increased': None,  # No comparison if VLAN is not in set1
                'old_tcn': None,
                'new_tcn': vlan_data_2['tcn']
            }

    return tcn_changes


if __name__ == "__main__":
    main()
```

في هذا المثال، ال Python نسخت نص تنفيذي إلى المفتاح يستعمل TFTP لاندل:

```
<#root>

Switch#

copy tftp://10.207.204.10/cat9k_tcn.py flash:/guest-share/cat9k_tcn.py

Accessing tftp://10.207.204.10/cat9k_tcn.py...
Loading cat9k_tcn.py from 10.207.204.10 (via GigabitEthernet0/0): !
[OK - 5739 bytes]


5739 bytes copied in 0.023 secs (249522 bytes/sec)
```

2. قم بتثبيت نص IM. في هذا المثال، المهمة الوحيدة لبرنامج IM النصي هي تشغيل البرنامج النصي Python، والذي يرسل رسالة سجل إذا تم اكتشاف زيادة TCN. في هذا المثال، يتم تشغيل نص IM كل 5 دقائق.

```
event manager applet tcn_monitor authorization bypass
 event timer watchdog time 300
 action 0000 syslog msg "TAC EEM-python script started."
 action 0005 cli command "enable"
 action 0015 cli command "guestshell run python3 /bootflash/guest-share/cat9k_tcn.py"
 action 0020 syslog msg "TAC EEM-python script finished."
```

3. للتحقق من وظائف البرنامج النصي، يمكنك إما تشغيل برنامج Python النصي يدويا أو الانتظار لمدة خمس دقائق حتى يدعوه برنامج IM النصي. في كلا الحالتين، أرسلت syslog فقط إن ال TCNs زادت ل VLAN.

```
<#root>

Switch#

more flash:/guest-share/tcns.json


{
"0001": {
"id_int": 1,
"tcn": 20,
"last_tcn": "00:01:18",
"source_interface": "TwentyFiveGigE1/0/5",
"tcn_in_last_day": true
},
"0010": {
"id_int": 10,
"tcn": 2,
"last_tcn": "00:00:22",
```

```
"source_interface": "TwentyFiveGigE1/0/1",
"tcn_in_last_day": true
},
"0020": {
"id_int": 20,
"tcn": 2,
"last_tcn": "00:01:07",
"source_interface": "TwentyFiveGigE1/0/2",
"tcn_in_last_day": true
},
"0030": {
"id_int": 30,
"tcn": 1,
"last_tcn": "00:01:18",
"source_interface": "TwentyFiveGigE1/0/3",
"tcn_in_last_day": true
}
}

Switch#
guestshell run python3 /flash/guest-share/cat9k_tcn.py

Switch#
*Feb 17 21:34:45.846: %GUESTSHELL-5-PYTHON_SCRIPT: Message from tty73(user id: shxUnknownTTY): TCN

Switch#
```

4. اختبر نص IM باناظار تشغيله كل خمس دقائق. إن زادت ال TCNs لأي VLAN، syslog رسالة أرسلت. في اذه المثال تحديدا، يالحظ أن شبكات TCN تتزايد باستمرار علي شبكة VLAN رقم 30، وأن الواجهة التي تتلقى شبكات TCN الثابتة هذه هي Twe1/0/3.

```
<#root>

*Feb 17 21:56:23.563: %HA_EM-6-LOG: tcn_monitor: TAC EEM-python script started.
*Feb 17 21:56:26.039: %GUESTSHELL-5-PYTHON_SCRIPT: Message from tty73(user id: shxUnknownTTY):

TCNs increased in VLAN 0030 from 3 to 5. Last TCN seen on TwentyFiveGigE1/0/3.

*Feb 17 21:56:26.585: %HA_EM-6-LOG: tcn_monitor: TAC EEM-python script finished.
*Feb 17 22:01:23.563: %HA_EM-6-LOG: tcn_monitor: TAC EEM-python script started.
*Feb 17 22:01:26.687: %HA_EM-6-LOG: tcn_monitor: TAC EEM-python script finished.
*Feb 17 22:06:23.564: %HA_EM-6-LOG: tcn_monitor: TAC EEM-python script started.
*Feb 17 22:06:26.200: %GUESTSHELL-5-PYTHON_SCRIPT: Message from tty73(user id: shxUnknownTTY):

TCNs increased in VLAN 0030 from 5 to 9. Last TCN seen on TwentyFiveGigE1/0/3.

*Feb 17 22:06:26.787: %HA_EM-6-LOG: tcn_monitor: TAC EEM-python script finished.
*Feb 17 22:11:23.564: %HA_EM-6-LOG: tcn_monitor: TAC EEM-python script started.
```

```
*Feb 17 22:11:26.079: %GUESTSHELL-5-PYTHON_SCRIPT: Message from tty73(user id: shxUnknownTTY):
TCNs increased in VLAN 0030 from 9 to 12. Last TCN seen on TwentyFiveGigE1/0/3.
*Feb 17 22:11:26.686: %HA_EM-6-LOG: tcn_monitor: TAC EEM-python script finished.
```

معلومات ذات صلة

- دليل تكوين قابلية البرمجة، Cisco IOS XE Cuteno 17.9.x (الفصل: Guest Shell)
- التعرف على أفضل الممارسات والنصوص المفيدة ل IM
- إستضافة التطبيق على التقرير الرسمي للموجهات من السلسلة Cisco Catalyst 9000 Series

حول هذه الترجمة
ترجمت Cisco هذا المستند باستخدام مجموعة من التقنيات الآلية والبشرية لتقديم محتوى دعم للمستخدمين في جميع أنحاء العالم بلغتهم الخاصة. يُرجى ملاحظة أن أفضل ترجمة آلية لن تكون دقيقة كما هو الحال مع الترجمة الاحترافية التي يقدمها مترجم محترف. تخلي Cisco Systems مسئوليتها عن دقة هذه الترجمات وتُوصي بالرجوع دائمًا إلى المستند الإنجليزي الأصلي (الرابط متوفر).