Fehlerbehebung: Bereitstellung und Anbindung von Secure Access Resource Connectors in Google Cloud Docker

Inhalt

Problem

Die Bereitstellung des Connectors für sichere Zugriffsressourcen im Docker war nicht erfolgreich.

Obwohl der Connector richtig installiert wurde, konnte keine Verbindung zu Cisco Secure Access hergestellt werden.

Diagnoseprüfungen haben einen Tunnelabbruch und Fehler bei der Serverkommunikation gemeldet.

Die Umgebung verwendet virtuelle Red Hat 9-Systeme, die in Google Cloud gehostet werden und über eine Fortinet-Firewall mit einer "any any"-Regel verbunden sind.

Die Fehlerbehebung hat potenzielle MTU-Diskrepanzen zwischen den Netzwerkschnittstellen als wichtigen Faktor aufgedeckt.

Umwelt

- Technologie: Solution Support (SSPT - Vertrag erforderlich)
- Subtechnologie: Sicherer Zugriff - Ressourcen-Connector (Installation, Upgrade, Registrierung, Konnektivität, private Ressource)
- Plattform: Red Hat 9 virtuelle Maschinen auf Google Cloud
- Netzwerk: Fortinet-Firewall zwischen sicherem Zugriff und VM (beliebige Regel vorhanden)
- Connector-Region: iuvz83r.mxc1.acgw.sse.cisco.com
- Google Cloud vPC-Standard-MTU: 1.460 Byte
- Docker-Bridge (docker0) Standard-MTU: 1.500 Byte (vor Änderung)
- Eine Netzwerkschnittstelle (eth0) pro VM

Auflösung

Gehen Sie folgendermaßen vor, um Verbindungsprobleme mit Secure Access Resource Connector in einer Docker-/Google Cloud-Umgebung zu diagnostizieren und zu beheben:

Überprüfen der DNS-Auflösung für die Connector-Region

Verwenden Sie `nslookup`, um zu bestätigen, dass der sichere Zugriffsbereich vom virtuellen System aufgelöst werden kann.

```
nslookup iuvz83r.mxc1.acgw.sse.cisco.com
```

Beispiel:

```
Server:         64.102.6.247
Address:        64.102.6.247#53
Non-authoritative answer:
Name:   iuvz83r.mxc1.acgw.sse.cisco.com
Address: 163.129.128.72
Name:   iuvz83r.mxc1.acgw.sse.cisco.com
Address: 163.129.128.70
Name:   iuvz83r.mxc1.acgw.sse.cisco.com
Address: 163.129.128.66
Name:   iuvz83r.mxc1.acgw.sse.cisco.com
Address: 163.129.128.68
```

Netzwerkverbindungen für sicheren Zugriff überprüfen

Verwenden Sie `Ping` und `Telnet`, um die Verbindung mit sicherem Zugriff vom virtuellen System aus zu validieren.

```
ping iuvz83r.mxc1.acgw.sse.cisco.com
```

Beispiel:

```
PING iuvz83r.mxc1.acgw.sse.cisco.com (163.129.128.66) 56(84) bytes of data.
64 bytes from 163.129.128.66: icmp_seq=1 ttl=57 time=44.7 ms
64 bytes from 163.129.128.66: icmp_seq=2 ttl=57 time=43.8 ms
...
telnet iuvz83r.mxc1.acgw.sse.cisco.com 443
```

Beispiel:

```
Trying 163.129.128.66...
Connected to iuvz83r.mxc1.acgw.sse.cisco.com.
Escape character is '^]'.
```

Überprüfen der Tunnelverbindung und Durchführen einer Diagnose

Führen Sie das Connector-Diagnosedienstprogramm aus, um den Tunnelstatus zu überprüfen.

```
/opt/connector/data/bin/diagnostic
```

Beispiel:

```
###check tunnel connection:
error: tunnel is not connected
```

Überprüfung der Netzwerkschnittstelle und der MTU-Einstellungen

Überprüfen Sie die IP-Adressen und die MTU aller Schnittstellen mithilfe von `ifconfig` und `ip a`.

```
ifconfig
ip a
```

Beispielausgabe für eth0 und docker0:

```
[root@degcpprcra02 ~]# ifconfig
docker0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet x.x.x.x netmask x.x.x.x broadcast x.x.x.x
inet6 fe80::1c66:46ff:fe1d:8bed prefixlen 64 scopeid 0x20<link>
ether 1e:66:46:1d:8b:ed txqueuelen 0 (Ethernet)
RX packets 974 bytes 119775 (116.9 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 848 bytes 161554 (157.7 KiB)
TX errors 0 dropped 2 overruns 0 carrier 0 collisions 0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1460
inet x.x.x.x netmask x.x.x.x broadcast 0.0.0.0
ether 42:01:c0:a8:80:b0 txqueuelen 1000 (Ethernet)
RX packets 20175 bytes 7755728 (7.3 MiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 21550 bytes 31402300 (29.9 MiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Überprüfen, ob TCP-Datenverkehr erfasst wird

Verwenden Sie `tcpdump`, um den Datenverkehr zwischen dem virtuellen System und der Secure

Access-Region zu erfassen.

```
tcpdump -i eth0 host iuvz83r.mxc1.acgw.sse.cisco.com
```

Beispielausgabe (ohne erfasste Pakete):

```
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
^C
0 packets captured
6 packets received by filter
0 packets dropped by kernel
```

Zerstören Sie den Anschluss, und installieren Sie ihn bei Bedarf neu.

Beenden und zerstören Sie den Steckverbinder, wenn Diagnose und technischer Support nicht funktionieren:

```
/opt/connector/install/connector.sh stop --destroy
cd /opt
rm -rf connector
```

Installieren Sie den Anschluss neu, und erzeugen Sie eine Ausgabe des technischen Supports.

Generieren Sie nach der Neuinstallation den technischen Support, um Fehlerprotokolle zu erfassen:

```
/opt/connector/data/bin/techsupport > techsupport.txt
Sample output showing connection errors:
2026-02-13 23:48:20.398772500  >> warning: Connection attempt has failed.
2026-02-13 23:48:20.398775500  >> warning: Unable to contact iuvz83r.mxc1.acgw.sse.cisco.com.
2026-02-13 23:48:20.398775500  >> error: Connection attempt has failed due to server communication erro
2026-02-13 23:48:20.398887500  >> state: Disconnected
```

Anpassen der Docker-MTU an die Google Cloud VPC- und VM-Schnittstelle

Ändern Sie die MTU auf der Docker-Bridge-Schnittstelle so, dass sie der Google Cloud VPC-Standardeinstellung entspricht (1460 Byte):

```
ip link set dev docker0 mtu 1460
```

Überprüfen Sie die MTU-Änderung:

```
ip a
```

Beispiel:

```
docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc noqueue state UP group default
link/ether 1e:66:46:1d:8b:ed brd ff:ff:ff:ff:ff:ff
inet x.x.x.x brd x.x.x.x scope global docker0
    valid_lft forever preferred_lft forever
inet6 fe80::1c66:46ff:fe1d:8bed/64 scope link
    valid_lft forever preferred_lft forever
```

Persist Docker MTU-Änderung in **`/etc/docker/daemon.json`**

Bearbeiten Sie `/etc/docker/daemon.json`, und fügen Sie den `mtu`-Wert hinzu bzw. aktualisieren Sie ihn:

```
{
  ...
  "mtu": 1460
}
```

Starten Sie das virtuelle System neu, um die MTU-Konfiguration anzuwenden.

Starten Sie das gesamte virtuelle System neu, um sicherzustellen, dass die MTU-Einstellungen vollständig angewendet werden. Dies ist erforderlich, da es möglich ist, dass ein Neustart nur des Docker-Diensts nicht die MTU-Änderung für alle Netzwerkkomponenten erzwingt.

Nachdem diese Schritte ausgeführt wurden, wurde die Verbindung mit sicherem Zugriff erfolgreich hergestellt, und die Konfiguration konnte abgeschlossen werden.

Ursache

Die Ursache war eine MTU-Diskrepanz zwischen der Docker-Bridge-Schnittstelle (docker0) und der Google Cloud VPC/VM-Netzwerkschnittstelle (eth0). Die Google Cloud VPC- und VM-Schnittstellen verwenden standardmäßig eine MTU von 1460 Byte, während die Docker-Standard-

MTU 1500 Byte beträgt.

Diese Diskrepanz verursachte eine Fragmentierung oder verworfene Pakete, wodurch der Secure Access Resource Connector keinen Tunnel erstellen konnte. Durch das Ausrichten der MTU-Werte wurde das Verbindungsproblem behoben.

Verwandte Inhalte

- https://securitydocs.cisco.com/docs/csa/olh/120695.dita
- https://securitydocs.cisco.com/docs/csa/olh/120776.dita
- https://securitydocs.cisco.com/docs/csa/olh/120727.dita
- https://securitydocs.cisco.com/docs/csa/olh/120772.dita
- https://securitydocs.cisco.com/docs/csa/olh/120762.dita
- https://securitydocs.cisco.com/docs/csa/olh/120685.dita
- Technischer Support und Downloads von Cisco

Informationen zu dieser Übersetzung

Cisco hat dieses Dokument maschinell übersetzen und von einem menschlichen Übersetzer editieren und korrigieren lassen, um unseren Benutzern auf der ganzen Welt Support-Inhalte in ihrer eigenen Sprache zu bieten. Bitte beachten Sie, dass selbst die beste maschinelle Übersetzung nicht so genau ist wie eine von einem professionellen Übersetzer angefertigte. Cisco Systems, Inc. übernimmt keine Haftung für die Richtigkeit dieser Übersetzungen und empfiehlt, immer das englische Originaldokument (siehe bereitgestellter Link) heranzuziehen.