Troubleshoot Login Issues to ASR5500 Due to Idle noTTY Sessions

Contents

Introduction

This document describes how to troubleshoot scenarios when Secure Shell (SSH) connectivity is lost to the Management IPs of the Aggregation Services Router (ASR5500/ASR 5000).

Login Issues to ASR5500 Nodes

You are unable to log in to ASR5500 Packet core nodes. The SSH connection immediately gets terminated without the login prompt. Telnet connections exhibit similar behaviour.

Steps to Troubleshoot

Step 1. Attempt to log in to the node through the console connection.

Step 2. In most cases, no specific Simple Network Management Protocol (SNMP) traps are issued that could point to the cause of the connection failure.

Step 3. The logs related to login, constantly present in the syslogs are:

```
evlogd: [local-60sec55.607] [tacacs+ 37201 error] [5/0/8908 <vpnmgr:1> authen_events.c:598]
[software internal system critical-info syslog] Authentication failed for user **** on tty
/dev/pts/0, application ssh, remote IP address XX.XX.XX.XX
evlogd: [local-60sec55.623] [cli 30028 debug] [5/0/8908 <vpnmgr:1> luser_auth.c:1448] [context:
local, contextID: 1]  [software internal system syslog] Login attempt failure for user **** IP
address XX.XX.XX.XX - Access type ssh/sftp
evlogd: [local-60sec53.652] [tacacs+ 37201 error] [5/0/8908 <vpnmgr:1> authen_events.c:598]
[software internal system critical-info syslog] Authentication failed for user **** on tty
/dev/pts/0, application ssh, remote IP address XX.XX.XX.XX
evlogd: [local-60sec53.679] [cli 30028 debug] [5/0/8908 <vpnmgr:1> luser_auth.c:1448] [context:
local, contextID: 1]  [software internal system syslog] Login attempt failure for user **** IP
address XX.XX.XX.XX - Access type ssh/sftp#####
evlogd: [local-60sec2.942] [tacacs+ 37201 error] [5/0/8908 <vpnmgr:1> authen_events.c:598]
[software internal system critical-info syslog] Authentication failed for user epcats on tty
/dev/pts/0, application ssh, remote IP address YY.YY.YY.YY
```

Step 4. The command **show crash list all** displays the recent crashes, note that the ones related to **vpnmgr** are especially important.

Step 5. The command **show task resources all** ensure that **vpnmgr** and **sshd** processes must not be in overstate. **vpnmgr** is responsible for IP address pool management and performs all context-specific operations. **sshd** supports secure login to the StarOS CLI.

Step 6. The restart of **vpnmgr** instance 1. helps to get SSH connection back with minimal impact in some cases. However, the connection might terminate after a while.

Step 7. The MIO switchover resolves the issue. Please note that in scenarios where a process might reach a threshold value or an overload state, MIO bounce can assist in order to clear it.

The workaround in place is MIO switchover. The next section talks about the steps for root cause analysis.

Root Cause Analysis

1. Use the **show administrators** command in order to determine the number of active connections on the node. However, the output might not exhibit an excessive number of active sessions which might have clogged the connections to the node.

Sample output:

```
[local]ASR5500-2# show administrators
Monday September 06 13:15:07 CDT 2021
Administrator/Operator Name    M Type    TTY            Start Time               Mode
Idle

------------------------------ - ------- -------------- ------------------------ ------------ --
--

admin                            admin   /dev/pts/4     Mon Sep 06 13:14:38 2021 Context User 29

admin                            admin   /dev/pts/3     Mon Sep 06 12:21:13 2021 Context User
749

admin                            admin   /dev/pts/2     Thu Sep 02 11:03:57 2021 Context User
342206
[local]ASR5500-2#
```

2. Further, execute these commands and dig into the issue. Navigate to the debug shell through the hidden mode.

```
cli test-command pass <password>
debug shell
```

Run these commands in the debug shell:

```
ps -ef
setvr 1 bash
netstat -n
```

ps - list processes. The **ps** command allows you to view technical information about current processes on a system as well as verify their status.

-e - show all processes, irrespective of the user.

-f - show processes in detailed format.

The **netstat** command is one of the most convenient command line options that is used to display all the socket connections that are present to the node. It possesses the capability to list out all the tcp and udp socket connections, as well as the unix connections. This CLI can also be used to list out the possible listening sockets that might still wait for a connection to get established.

Sample output:

```
ASR5500-2:card5-cpu0# ps -eF

UID        PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD

root         1     0  0   511   640   4 Aug20 ?        00:00:13 init [5]

root         2     0  0     0     0   2 Aug20 ?        00:00:00 [kthreadd]

root         3     2  0     0     0   0 Aug20 ?        00:00:00 [ksoftirqd/0]

root         6     2  0     0     0   0 Aug20 ?        00:00:00 [migration/0]

root         7     2  0     0     0   0 Aug20 ?        00:00:01 [watchdog/0]

root         8     2  0     0     0   1 Aug20 ?        00:00:00 [migration/1]

root        10     2  0     0     0   1 Aug20 ?        00:00:00 [ksoftirqd/1]

root        11     2  0     0     0   0 Aug20 ?        00:00:31 [kworker/0:1]

root        12     2  0     0     0   1 Aug20 ?        00:00:00 [watchdog/1]

root        13     2  0     0     0   2 Aug20 ?        00:00:00 [migration/2]

root        15     2  0     0     0   2 Aug20 ?        00:00:00 [ksoftirqd/2]

root        16     2  0     0     0   2 Aug20 ?        00:00:00 [watchdog/2]

root        17     2  0     0     0   3 Aug20 ?        00:00:00 [migration/3]

root        19     2  0     0     0   3 Aug20 ?        00:00:00 [ksoftirqd/3]

root        20     2  0     0     0   3 Aug20 ?        00:00:00 [watchdog/3]

root        21     2  0     0     0   4 Aug20 ?        00:00:00 [migration/4]

root        22     2  0     0     0   4 Aug20 ?        00:00:00 [kworker/4:0]

root        23     2  0     0     0   4 Aug20 ?        00:00:00 [ksoftirqd/4]

……..

ASR5500-2:card5-cpu0# setvr 1 bash
bash-2.05b# netstat -n

Active Internet connections (w/o servers)

Proto Recv-Q Send-Q Local Address           Foreign Address         State

tcp        0      0 10.201.211.23:22        10.227.230.222:51781    ESTABLISHED
```

```
tcp        0      0 10.201.211.23:22        10.24.28.55:49918       ESTABLISHED

tcp        0      0 10.201.211.23:22        10.99.10.148:54915      ESTABLISHED

tcp        0      0 10.201.211.23:22        10.227.230.222:51783    ESTABLISHED

Active UNIX domain sockets (w/o servers)

Proto RefCnt Flags       Type       State         I-Node Path

unix  2      [ ]         DGRAM                    39221385

unix  2      [ ]         DGRAM                     27056

bash-2.05b# exit
```

According to the previously mentioned report, servers ran scripts that spawned connections to the ASR55K box. These servers opened a lot of these connections that were either in a stuck or idle state, but they were never closed.

Even after the TeleTypeWriter (TTY) connection was terminated, the TCP connection remained active on our gateways.

As a result of these connections, the ASR5500 reached the maximum number of allowed SSH connections, obstructing the connection to the box. As soon as you try to log in to the servers and kill the parent processes, all connections are instantly released, and the SSH gets immediately restored.

These idle SSH connections get established as no TeleTypeWriter (noTTY) connections. Such noTTY connections are used by programs that are connected in such a way that their output is not displayed.

Commands such as SSH admin@asr55k hostname "display version" establishes a noTTY connection in most cases.

Similarly, statements as SSH: *@notty indicate that there are SSH logins to our Gateways (GWs) that haven't been assigned a visual terminal, such as a shell or pseudo-terminal. This can occur during a variety of script-related operations, particularly when using FTP/Secure Copy (SCP) connections.

Solution Proposed

1. Implement a timeout on the scripts which may be used for the API servers. Multiple SSH connections that execute multiple CLIs can generate messenger congestion and significant CPU usage on all sessmgr processes.
2. In order to make troubleshooting easier, configure this option:

```
logging filter runtime facility cli level debug critical-info
```

3. Apply this configuration to the node. This command is used to terminate idle SSH sessions after 5 minutes. This is used as a protection mechanism against stale sessions caused by the server:

```
Exec > Global Configuration > Context Configuration
configure > context context_name
administrator encrypted password timeout-min-absolute 300 timeout-min-idle 300
```

Related information

- CLI information
- Cisco ASR 5000 Series Configuration Guides
- **Technical Support & Documentation - Cisco Systems**