Arista EOS Weird Machine

Introduction

As a cyber operations student and network engineer who has the opportunity to support a large enterprise network environment, this research project on Weird Machines has given me new insights into behaviors I previously overlooked.

One of my responsibilities as a network engineer is to support ongoing network operations. It was during this work that I believe I discovered a Weird Machine, which I describe below.

Background on Machine

In our network we use Arista switches, which are a modern Multi-Layer Switch networking device that runs an "Arista Extensible Operating System" (EOS). Arista EOS is a Linux-based operating system that supports the functionalities of a modern Multi-Layer Switch (routing, switching, monitoring, logging, security, etc). While supporting operations one day, I noticed what now seems to be an emergent behavior of the Arista switch during normal operations that fits the definition of a Weird Machine. A Weird Machine at a high level to me is an unintended program or computational behavior arising from a normal, composed system or architectural components.

Our EOS switch configuration was configured to forward syslog messages to a remote server using the following configuration:

"logging vrf MGMT host 10.10.10.10 (fake-ip) protocol tcp"

The IP address we had configured on the switch was incorrect; this led to it being unreachable to the switch which triggered a chain reaction that was not intended by the developers of the EOS.

Expected behavior:

- When remote logging fails, local device logging continues

Local Logging is the storage of system and event logs directly on the originating device for on-box monitoring and troubleshooting. Remote logging is sending that devices' system and event logs to a remote server, so that they are stored off the box. Remote logging could fail due to many reasons; in this case we configured an unreachable / incorrect IP to send syslog to, causing a failure of remote logging.

Observed Behavior:

- Remote logging failed, but so did local logging. Local logging stopped entirely-no new entries appeared when I checked the switch's local log buffer using a "show log" command. Until the configured Ip was corrected to a reachable and correct remote syslog server.

Investigation

Working with Arista's Technical Assistance Center (TAC) to investigate and determine the root cause of the local logging failure was critical; it was reported from the TAC that Arista EOS uses the open-source Linux utility "rsyslog" as its system logging daemon. Which was confirmed through troubleshooting and checking the daemon status using "systemctl status rsyslog". EOS's default rsyslog configuration (checked via /etc/rsyslog.conf) forwarding actions used:

- "action(type=omfwd" target="10.10.10.10" protocol="tcp" port"514")

But lacked any specific queue parameters such as:

- Queue.type, queue.size, etc

This is important because that means TCP forwarding ran in direct or synchronous mode, not asynchronous mode. According to official rsyslog documentation and USENIX best practices, lack of an action queue for TCP forwarding causes blocking when the remote host is unreachable, halting local message processing (local logging).

Best Practices according to Rsyslog and USENIX documentation:

Rsyslog official: "Enable queues for TCP forwarding: Always define a queue (queue.type="linkedList") to avoid blocking if the remote server is unavailable.

USENIX "Solving Rsyslog Performance Issues": "For example, this configuration is dependent on a remote system; if that system is down, no local logs will be processed and we will see logs queue up and eventually fill the queue, causing programs trying to write logs to block. If we want to continue logging locally, even if the remote system is down, we can create a default-sized action queue for the TCP output action"

Root Cause

Arista EOS's rsyslog configuration used a block TCP path with no decoupling queue, unintentionally tying local logging to remote network state.

Weird Machine Interpretation

The intended machine has independent local and remote logging paths; local logs should persist even if remote forwarding fails. However, the unintended machine or Weird Machine created a new control path that emerged where the remote TCP state directly controlled the local logging subsystem of the machine's architecture. The trigger was an unreachable remote syslog IP and the primitive interaction arose from rsyslog blocking TCP connection attempts within EOS's single threaded logging path. The emergent behavior completely halted local logging caused by remote unreachability, a coupling that should not logically exist. The interaction between TCP transport, rsyslog's queue model, and Arista EOS's overall logging architecture created an emergent machine that executed an unintended program unforeseen by designers: **stop all logging until TCP succeeds.** EOS's internal design effectively transformed network reachability into a control condition for local logging. This highlights the importance of system composition awareness and fault isolation in complex architecture. Finally, Weird Machines are not necessarily rare but are inevitable and recognizing them helps improve resilience and trust in system/architectural design.

Sources:

Rsyslog Official Documentation:
https://www.rsyslog.com/doc/configuration/modules/omfwd.html

USENIX "Solving Rsyslog Performance Issues":
https://www.usenix.org/system/files/login/articles/06_lang.pdf