Embedded Event Manager (EEM)

Applets for Routers

Presented by:
Jason Bomar, CCIE #9316

Introduction

We will cover a number of topics, at a high level they are:

- Overview of EEM
- Scenarios
- Demo
- Applets
- References

Overview of EEM

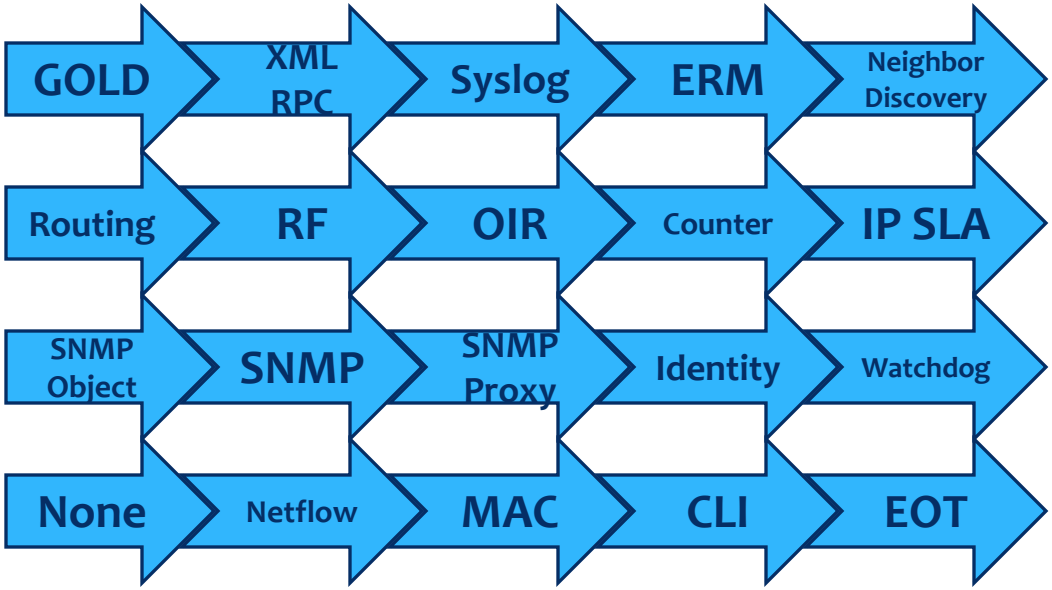
- Embedded – EEM is built into IOS and NX-OS, it has been a key feature of the systems for many years and becomes more feature rich with each new release.
- Event – Something that occurs on the network. It could be a link transitioning up/down, or it could be a device that is/is not reachable or a counter incrementing on a “show” command.
- Manager – An action that is taken when the “Event” occurs. What kind of action? It could be an IOS command, a customized syslog message, an email – or any of a number of things.

Overview of EEM

- EEM Event Detector – These are the actual events that can be detected against – there are loads of them
- EEM Server Subsystem – jlkjlkhlkjhlkjh
- EEM Policies – What are the actions you want the subsystem to take for the given event detector.

Overview of EEM

What events can you detect on? Plenty of them!



Overview of EEM

EEM configuration can come in two flavors: Applets and Scripts. We will NOT cover scripts. Applets you can "?" you way through, applets you need a deeper understanding of TCL and how to program/script in it. For applets, there are three basic steps:

- Applet name
- Event information – how to detect, what to look for
- Action to take – CLI to enter, Syslog to send, etc.

It is stored as part of the running configuration – it is fairly easy to review and find the applet with a simple "show run".

Scenarios

Scenario #1

One of the application teams has a process that runs in the night. Many times it is fine, but once in a while it fails. Since it does not work, they have determined it must be the network. You look at your SNMP server and although the WAN seems busy, it does not seem congested. You know that you have had issues with your SNMP in the past however where it only checks every five minutes. How can you look at the smaller window that SNMP might not get and see if there is a network issue?

Scenarios

Scenario #2

You roll into work one morning to find a down circuit. After opening a ticket with the ISP, you start doing some deeper looking into the issue and realize that there are a high number of CRC errors and a number of interface resets – but you have no way of knowing if they are somehow related to the circuit issue, or if they occurred too far in the past. If only you had some way to monitor when a specific counter/s from “show interface” incremented …

Scenarios

Scenario #3

Many nights sometime between 11:00 PM – 3:30 AM the CPU on your router goes through the roof. Due to the inconsistent nature of the issue, and the fact that it gets in the way of your ~~social life~~ Matrix marathon, you are not really sure what causes it. You feel likely that it is a process that is running but have no evidence. If only there were a way to capture what processes are running on the router in real time WHEN the CPU spike occurs!

Applets

//* This first script is used to detect and alert (via Syslog) a high TX load OR a high RX load as measured by the load of the specific interface itself. If the exit value (low threshold) is false (meaning there IS high load) then it sends the "RED ALERT" message. When it clears (the load drops to 40/255 ~15%), it sends the GREEN ALERT. We trigger on the RED ALERT so we do not continuously spam our syslog with "everything is good" messages. This is superior to normal SNMP because it can capture and alert you to "micro bursts" that happen within the 5-minute polling cycle that would normally not get recorded. In a lab scenario, you can test this by changing the interface bandwidth to 64Kb and then going to an attached router and sending a few thousand large pings. *//

Applets

event manager applet TX-RX_LOAD_HIGH

event tag HIGH-RX interface name FastEthernet0/1 parameter rxload entry-op gt entry-val 127 entry-type value exit-op lt exit-val 40 exit-type value exit-event true poll-interval 1

event tag HIGH-TX interface name FastEthernet0/1 parameter txload entry-op gt entry-val 127 entry-type value exit-op lt exit-val 40 exit-type value exit-event true poll-interval 1

trigger

correlate event HIGH-RX or event HIGH-TX

action 1.0 if $_interface_exit_event eq 0

action 1.1.1 syslog priority alerts msg "RED ALERT: $_interface_name $_interface_parameter is $_interface_value!!"

action 1.2 else

action 1.3.1 syslog priority informational msg "GREEN ALERT: $_interface_name $_interface_parameter is $_interface_value"

action 1.4 end

Applets

//* This script is used to proactively alert when certain interface errors occur - namely input errors, CRC errors OR interface resets INCREMENT. So we don't want aged error counters to trigger this alert, it should only alert us if the counter increments in a 60-second window. In this case we have sent a custom "YELLOW ALERT" but you could just as easily send it as a RED depending on how you word the message that is sent. The actions for changing an interface description are there simply as part of a lab - to prove that you got your script to work. You can add up to SIX Event Detectors to an applet. Here we are adding three (input, crc and resets). *//

Applets

event manager applet INF_ERRORS

event tag CRC interface name FastEthernet0/0 parameter input_errors_crc entry-op ge entry-val 0 entry-type increment poll-interval 60

event tag INPUT interface name FastEthernet0/0 parameter input_errors entry-op ge entry-val 0 entry-type increment poll-interval 60

event tag RESETS interface name FastEthernet0/0 parameter interface_resets entry-op ge entry-val 0 entry-type increment poll-interval 60

trigger occurs 1

correlate event CRC or event INPUT or event RESETS

action 1.0 cli command "enable"

action 1.1 cli command "config t"

action 1.2 cli command "int fa0/0"

action 1.3 cli command "description EEM script worked!"

action 2.0 syslog priority alerts msg "YELLOW ALERT: $_interface_name $_interface_parameter has incremented $_interface_value!!"

Applets

//* This is a cool, but more advanced script. Two things cause a CPU to spike - either a process or traffic. If the spike occurs off hours, or you are forced to respond to the alert a few hours (or one hour even) after it happened, it might well be impossible to determine WHY it spiked. Here we run commands in REAL TIME that capture this information and store it on flash - even if the router crashed you could feed this info to TAC for analysis. Also, since what we are detecting here is a rising CPU level, even though protections occur for EEM and CPU's, once we run this configuration, we will remove the applet so it has no chance of causing any harm. the OID here corresponds to a 5-sec CPU timer and there is a nice tool on the EEM community for looking these OID's up. You could walk in @ 8:00 AM, grep the syslog for RED ALERTS, see this, SSH to the router and TFTP the files down to your laptop and investigate! *//

Applets

```
event manager applet CPU-TSHOOT
 event snmp oid 1.3.6.1.4.1.9.9.109.1.1.1.1.6 get-type next entry-op gt entry-val "65" poll-interval 5
 action 0.1 cli command "enable"
 action 0.2 syslog priority alerts msg "RED ALERT: Capturing high cpu information to flash:"
 action 0.3 cli command "term length 0"
 action 1.1 cli command "show process cpu sort | append flash:EEM_CPU"
 action 1.4 cli command "show ip flow top | append flash:EEM_TRAFFIC"
 action 2.1 syslog priority alerts msg "Removing EEM APPLET from running_config"
 action 3.1 cli command "configure terminal"
 action 3.2 cli command "no event manager applet CPU-TSHOOT"
 action 3.3 cli command "end"
 action 3.4 cli command "term default length"
```

Resources

- Support forums for this technology are GREAT
- They even have videos!
- How about a good intro lesson to applets?
- Still not enough? This guy knows a thing or two…

Thank You!