Contents

Introduction

This document describes the Throttling of AAA (RADIUS) records feature which supports throttling of access (authentication and authorization) and accounting records that are sent to the RADIUS server.

This feature allows a user to configure the appropriate throttling rate to avoid network congestion and instability when there is insufficient bandwidth to accommodate a suddenburst of records generated from the Cisco router to the RADIUS server.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

The information in this document is based on ASR5k platfrom.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, make sure that you understand the potential impact of any command.

Background Information

When aaamgr sends the radius messages to RADIUS server at a high rate (e.g. when a large number of sessions goes down at the same time, accounting stop messages for all the sessions are generated at the same time) the RADIUS server may not be able to receive the messages at such high rates. To handle this condition we need an effective rate control mechanism at aaamgr, so that aaamgr sends messages at an optimal rate such that RADIUS server is capable of receiving all the messages and ensures that no message is dropped due to over load at the RADIUS server.

Mechanism of work

When aaamgr sends messages at the configured rate to the RADIUS server, it sends messages evenly throughout
each second rather than sending all the messages in a single burst. Depending on the configuration, each second is divided into several equal time slots (with a specific time period per slot). The minimum time-period of a slot
could be 50 milliseconds.

The rate has to be configured taking these into account

- The rate of incoming calls,
- Number of aaamgr instances
- The rate at which the RADIUS server can receive the messages and
- Interval of Interims (for accounting configuration)
- Algorithm used for server selection

If the configured value for authentication servers value is too low, then there will be a bottle neck leading to
congestion, which may lead to calls getting dropped due to session setup timeout. If a low value is configured for accounting servers, then a lot of purging of accounting messages will be observed, due to overflow of the queue.

When the feature is configured, the number of time slots in a second and time period of a second are computed and stored at the radius level. When a message is ready to be sent to RADIUS server, it is checked whether the quota (number of messages for this time slot) has reached. If the limit is not reached, the message is sent, if it is, then the message is queued in the server level queue to be sent in future time slots. Each RADIUS server holds details about number of messages sent in the current time slot and the time at which the time slot expires. When the queued messages are picked from the server level queue, they are put in the head of the instance level queue, ensuring preference for older messages than any other new message. Messages from the instance level queue are picked for servicing.

AAAMGR queues

There are two types of queues at AAAMGR for messages:

1. Instance level queues
2. Server level queues

When a message is generated, it is initially queued in the instance level queue for servicing.

The instance level queue is processed for 25 milliseconds for every 50 milliseconds. Any message that is dequeued from the instance level queue will be attempted to be sent to the RADIUS server. Under some conditions we may not be able to send the messages (no available band-width or no available IDs). In such cases, the messages that failed the attempt will be queued in the server level queues. For every 50 millisecond you pick as many messages that have IDs available and also band-width available and put them at the head of the instance level queue (these messages are older than any other message that is present in the instance level queue).

When there is a rate control for accounting messages, and if there are a lot of accounting messages in the instance level queue, then any new authentication message goes to the tail of the instance level queue. For getting processed it has to wait for all the accounting message (preceding the new auth message) to be either sent to RADIUS server or to be moved to server level queue. It is an existing behavior and it is not modified. So it can cause a small delay for the new auth message to get processed.

Example

Based on max-rate with value of 5, you can send five messages in 1 second and have 256 radius authentication messages outstanding (default max-outstanding configuration) unanswered per aaamgr towards Radius authentication server. In case there are more than 5 messages, in 1 second the messages are put in queue till AAA server responds to the existing requests.

In case you reach 256 Radius authentication messages sent from one aaamgr towards server, remaining requests will be put in queue till AAA server responds to the existing requests. It will again go into the same queue as that of max-rate. The message are picked up from the queue only when you have a free slot. The free slot comes in when you receive a response for the message or when it times-out.

Limitations

Since Cisco ASR5K is a distributed system with independent sessmgr/aaamgr pairs processing the calls, the rate throttling could be implemented only for independent aaamgr instances. It is theoretical to extend the rate of a single instance to the entire Cisco ASR5K box as a whole by just multiplying the total number of instances with the max-rate
of each instance.

This number is just the absolute upper limit in a sunny day scenario. You cannot treat Cisco ASR5K as a black-box and cannot assume that the all calls should succeed if the calculated value seen in the system did not cross the upper limit.

Radius max-rate is tied with other internal and external parameters related to the system. Please see the expected impact if one of the conditions is not met.

Conditions	**Impact if not met with**
Uniform distribution of calls from demuxmgr to all the sessmgrs	If the call distribution is not uniform, then radius messages be queued for some instances. So even though theoretica max-rate limit is not reached, calls will be dropped for inst where messages are queued.
Uniform distribution of IMSI's (this is just in case of round-robin mediation accounting)	Mediation-accounting round-robin is based on IMSI-based routing. In this case, based on the IMSI distribution, some set of servers may be preferred over others based on the routing logic, queue might be built-up for those servers leading to drop.
No sudden bursts of calls coming-in	If there is a burst of new calls, then again the newly gener radius messages will be queued in the system. By the tim new radius requests are processed. The session setup tim may be expired leading to call drops.
Radius servers should respond in time	When radius requests times-out because of server issues there'll again be queue build-up, because new requests w

be sent out unless the current one expecting a response i
removed from the system. The rate at which the timed-ou
messages will be removed from the system is dependent
max-outstanding and time-out configurations as well.

In many cases we can see that access requests are not processed by all the active aaamgr tasks. That means we are having uneven call distribution within the sessmgr tasks and further on, not all aaamgr instances are involved into call processing.

Call distribution is not based on the strict round robin mechanism that is if there are 10 incoming calls they will go to 10 sessmgrs in a monotonic algorithm.

Call distribution is based on these four main factors

- **active_session_count**
- **cpu_load**
- **Round_trip_delay** (demuxmgr – sessmgr – demuxmgr)
- **outstanding_add_request** (demux to sessmgr)

This is the current implementation. The max-rate is just an upper limit, but because of distributed nature of our architecture, you can’t directly extrapolate it to the chassis load. The behavior depends on load on a given AAAmgr
at a given time.

Radius max-rate queue should be used to **monitor** the **status** of the system. If there is a **queue build-up**, then
it means one of these 4 (refer to the table) conditions is not met with and you must identify the root cause for the same.

**max-rate queue threshold could be implemented and constantly monitored.