Guardium Administration

Second Installment - Part I & II.1 -

Maintaining a Balanced Environment

Handling Overloaded Collectors

#1 : Why Do Guardium Collectors Get Under Stress ?

# **Database Traffic is hectic by nature** and no one controls it. Therefore Guardium teams need to adapt to it.

# **Hectic traffic does put stress on appliances.** Here are the 4 major ones:

# **Signs of Hyper Variation of Traffic : Spikes**

- Large Variations on Eth0 Rec., Analyzer Rate, Analyzer Queue Length
- Spikes on increases in MySQL Disk Usage

# **Signs of Unbalanced Traffic : Overloads**

- Large differences among appliances on Eth0 Rec, Analyzer Rate, Logger Rate
- Large differences among appliances on System CPU Loads, MySQL Disk Usage

# **Signs of Reaching the Limits : in your Red Zone**

- Sniffer Memory close to ⅓ of total memory
- Mysql Disk Usage close to 90%

# **Signs of Being Beyond the Limits: you got outscored**

- Sniffer restarts frequently (many times a day)
- MySQL has reached 90% and the sniffer is down

Author: Frederic Petit

#3.1 : How to Detect and Handle Unbalanced Collectors

# What is Unbalanced Traffic ?

- Simply speaking : some collectors are **overloaded** , other **underloaded**, or in other words, some received, **overall** way too much traffic as compared to others

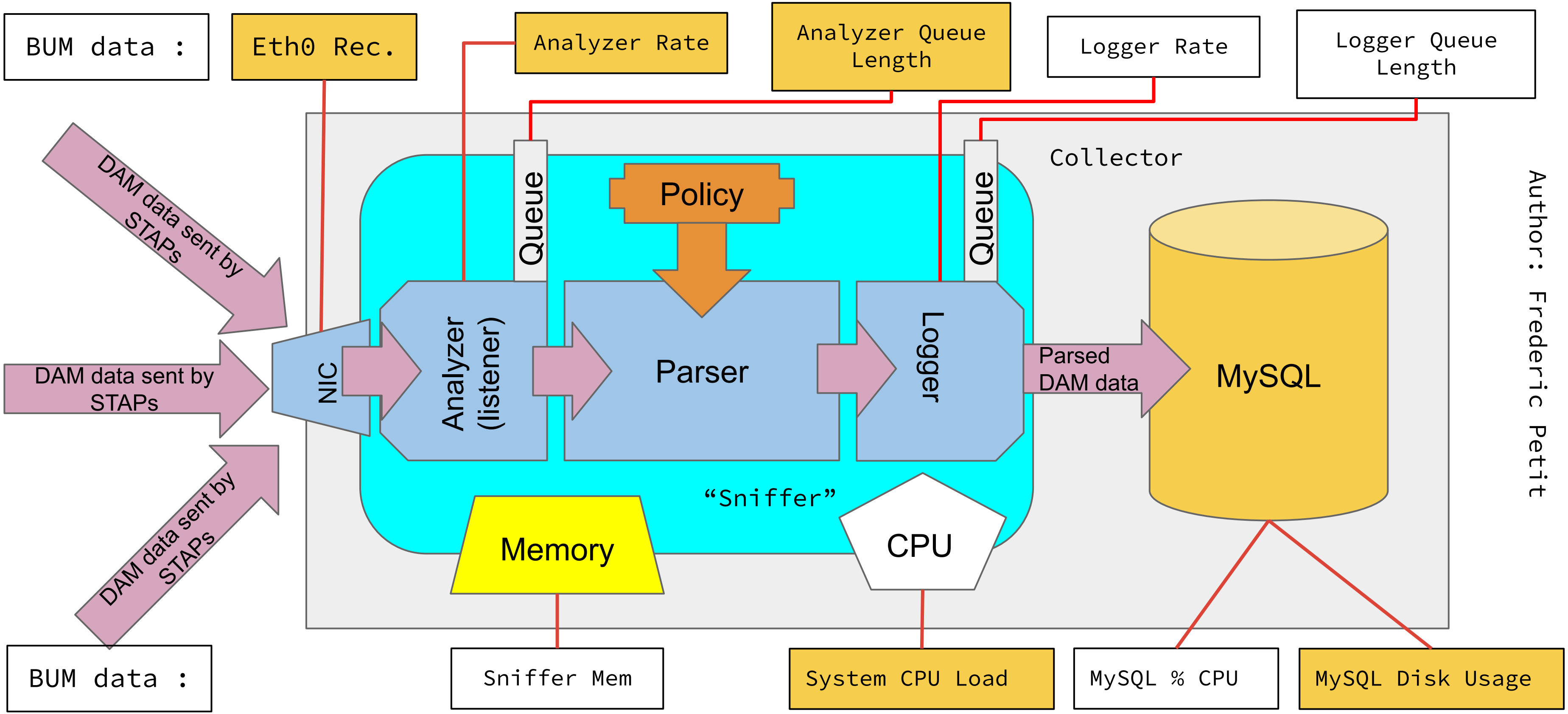
# Why to watch for Overloaded vs. Underloaded Collectors ?

- Overloaded collectors tend to get into trouble and will require maintenance, which Guardium teams should avoid having to do
- It can degrade very quickly. Therefore you should be pro-active as being re-active may already be too late

# How to detect and handle Overloaded vs. Underloaded Collectors ?

- Step #1 :
  - Monitor BUM variables Eth0 received, Analyzer rate, Analyzer Queue Rate, System CPU Load, MySQL Disk Usage
  - Compare the performances between collectors by ranking (see demo)
- Stop #2 :
  - Assess the contribution of each Agents on the Collector

Guardium Collectors Internal Architecture and the BUM



Author: Frederic Petit

Demo

#3.2 : How to Detect Contributing Servers/Agents

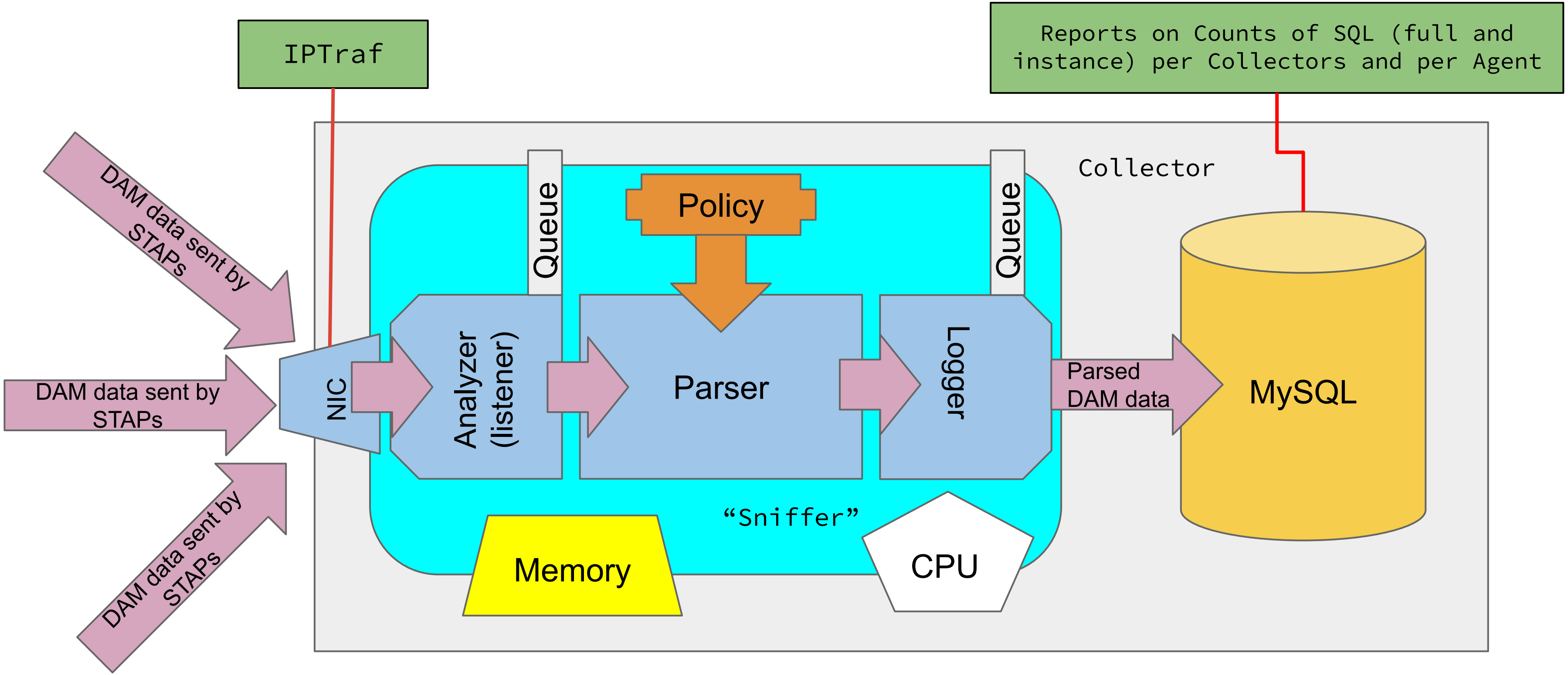
# **How to assess** the contribution of each Agent ?

- **Unfortunately** the BUM gives ONLY global statistics -
- **Only 2 places : NIC and MySQL (see diagram)**
  - At the NIC level : IPTraf from CLI
  - At the MySQL Level : Statistical Reports counting the number of SQLs

# **What to do ?**

- Do nothing is **rarely an option** in this case
- **Re-Assign** some Agents to underloaded collectors to reach a more balanced environment
- **Potentially** Activate the **E**nterprise **L**oad **B**alancing, but be careful, this too requires close monitoring and speedy reaction in case of trouble

# 3.2 : Assessing Contributing Agents (Not in the BUM)



IPTraf

Reports on Counts of SQL (full and instance) per Collectors and per Agent

DAM data sent by STAPs

DAM data sent by STAPs

DAM data sent by STAPs

Collector

NIC

Queue

Analyzer (listener)

Policy

Parser

Queue

Logger

Parsed DAM data

MySQL

"Sniffer"

Memory

CPU

Author: Frederic Petit

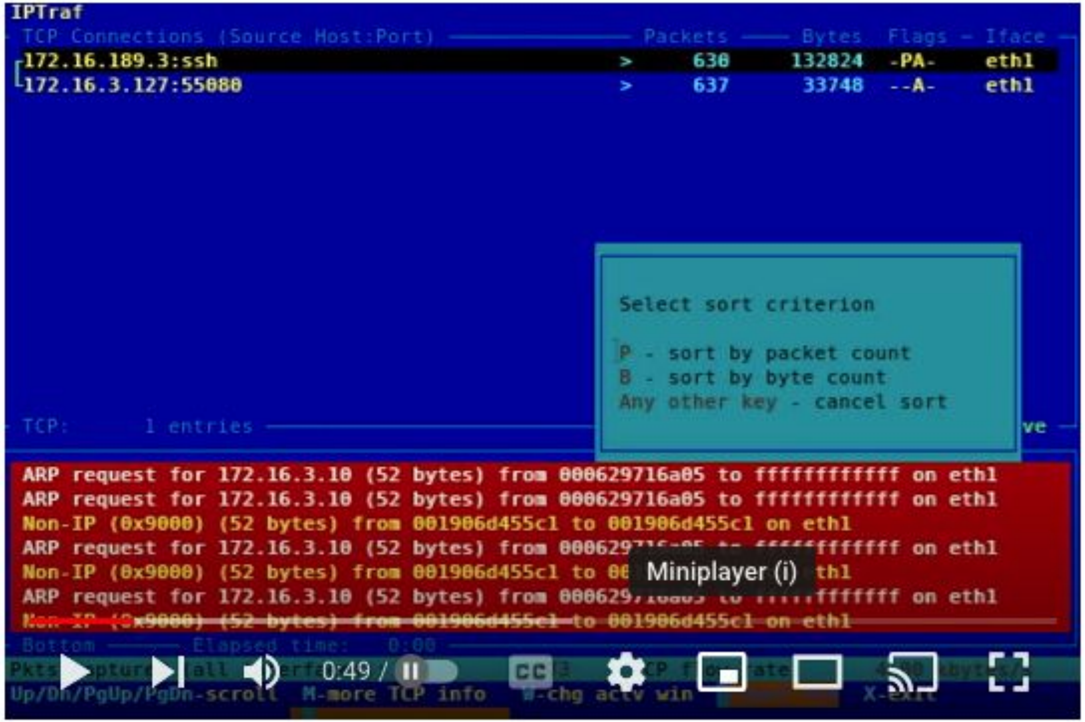
Option #1 : IPTraf

In CLI, just type in :

```
>iptraf
```

Excellent Tutorial video on IPTraf

https://youtu.be/D91hg8sEcOw



```
IPTraf
 TCP Connections (Source Host:Port) ───────────── Packets ──── Bytes  Flags - Iface ─
┌172.16.189.3:ssh                                  >     638     132824  -PA-    eth1
└172.16.3.127:55080                                >     637      33748  --A-    eth1

                                                   Select sort criterion

                                                   P - sort by packet count
                                                   B - sort by byte count
                                                   Any other key - cancel sort
 TCP:        1 entries ─────────────────────────────                             ve ─
 ARP request for 172.16.3.10 (52 bytes) from 000629716a05 to ffffffffffff on eth1
 ARP request for 172.16.3.10 (52 bytes) from 000629716a05 to ffffffffffff on eth1
 Non-IP (0x9000) (52 bytes) from 001906d455c1 to 001906d455c1 on eth1
 ARP request for 172.16.3.10 (52 bytes) from 00062971                ffffff on eth1
 Non-IP (0x9000) (52 bytes) from 001906d455c1 to 00                   th1
 ARP request for 172.16.3.10 (52 bytes) from 000629                   ffffff on eth1
 Non-IP (0x9000) (52 bytes) from 001906d455c1 to 001906d455c1 on eth1
 Bottom ──── Elapsed time:   0:00
 Up/Dn/PgUp/PgDn-scroll  M-more TCP info   W-chg actv win          X-exit
```

Option #2 : SQLs Recorded into MySQL - By Product of DAM

This is the tricky part :

- Requires having centralized/concentrated the DAM data into an ELK instance
- Requires to move the DAM Traffic from the Collector to an ELK instance

Our Solution to move DAM Data from Collector to an ELK instance:

- Export only PART of the DAM Traffic and send them to a Central ELK instance thru the **CT22T Enrichment process**
- Leaves open the possibility to keep the Aggregators

Very large topic we Keep for another video and presentation

See You In the next Video

On my YouTube Channel

Context22