Protect 2014

Washington, D.C. September 8-11

# Correlating efficiently

Rob Block

Lead Engineer, ArcSight Correlation

# Agenda

- Introduction
- Filters
- Real time correlation
- Reporting
- Trends to rescue
- Q & A

# Introduction

# Correlating efficiently: Goals

- Understand performance impact of the content you author
- Choose between different ways to solve a problem based on efficiency
- If you must write costly (performance wise) content, how can you mitigate impact on the system
- How to troubleshoot content related performance problems

# Rare resources

## Memory consumption

- Lot of memory—where did it all go?

## CPU consumption

- Where are all the CPU cycles being spent?

## Database hits

- Who is making Oracle do so much work?

# Filters

# Filtering efficiently

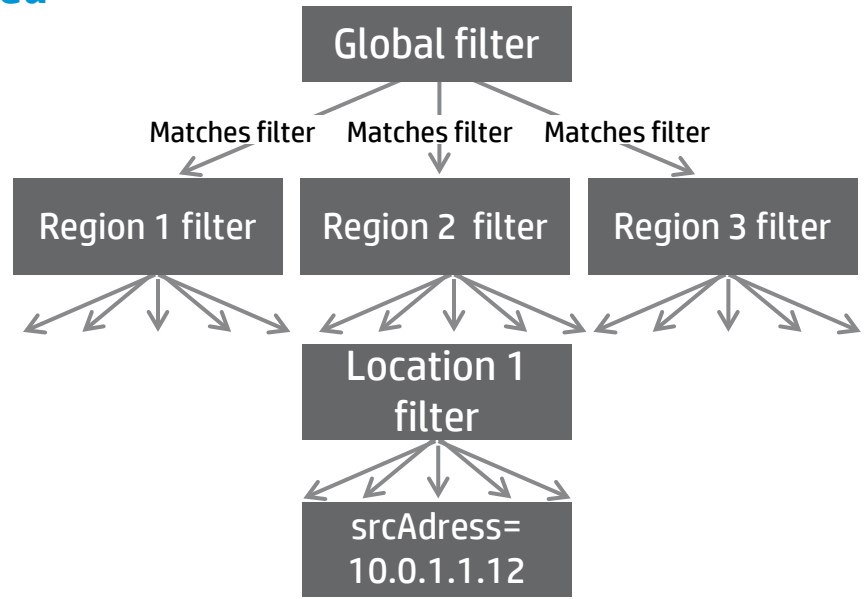
Filters: a double-edged sword

- Hit the database when used in reports/channels
- Consume CPU cycles when used in
  - Rules
  - Threat prioritization
  - Data monitors
- Minimal memory consumption

# Inefficient filter tree

## A global filter that restricts the view to events from interesting sources

## Hierarchically organized



# A more efficient alternative

- Build an active list containing all the IP addresses
- Filter is simply an inActiveList condition
- Advantages
  - Faster evaluation
  - Easier to debug
- Disadvantage
  - Can't use it in channels anymore **but** can use it in Query viewers
- Another alternative: Asset Modeling

# Filter debugging

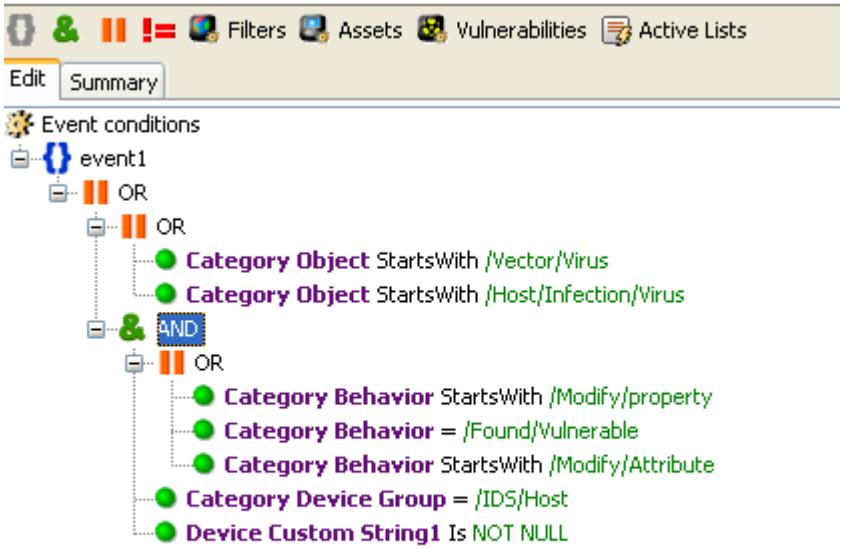
Debug which filter condition is not satisfied by the event

**Example:**

A filter is being used in a rule, an expected event should trigger the rule

**Problem:**

Rule did not get triggered by the expected event

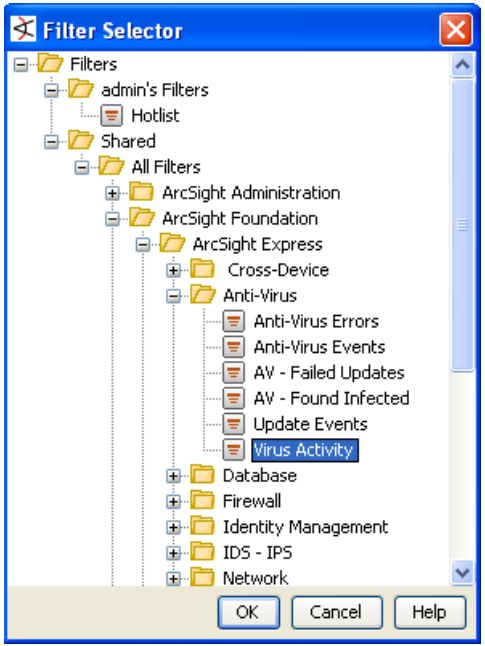


# Filter debugging

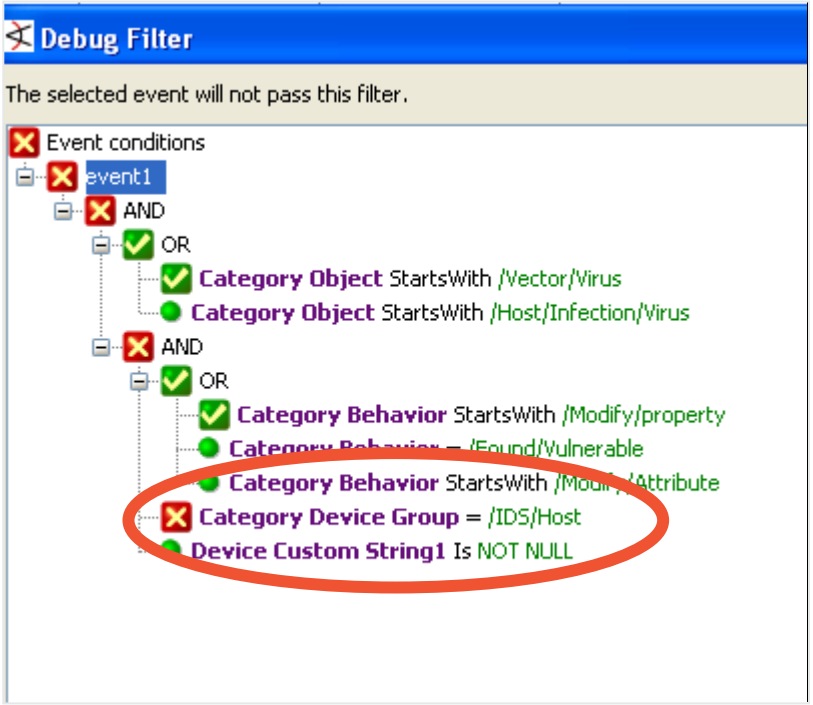
## Solution: Debug which filter conditions did not match the event

| Name | Category Object | Category Behavior | Category Device Group | Device Custom String1 |
|---|---|---|---|---|
| Top value count data monit… | /Host/Application | /Execute/Response | /Security Information M… | Unknown |
| Top value count data monit… | /Host/Application | /Execute/Response | /Security Information M… | Virus Activity |
| Virus Activity | /Vector/Virus | /Modify/property | /IDS/Host/Antivirus | suspicious activity |
| Virus Activity | /Vector/Virus | /Mo | | icious activity |
| Virus Activity | /Vector/Virus | /Mo | | icious activity |
| Top value count data monit… | /Host/Application | /Ex | | ource/Update |
| Top value count data monit… | /Host/Application | /Ex | | ource/Add |
| Top value count data monit… | /Host/Application | /Ex | | ource/Add |
| Filter updated | /Host/Application | /Mo | | |
| Top value count data monit… | /Host/Application | /Ex | | ager Internal Age… |





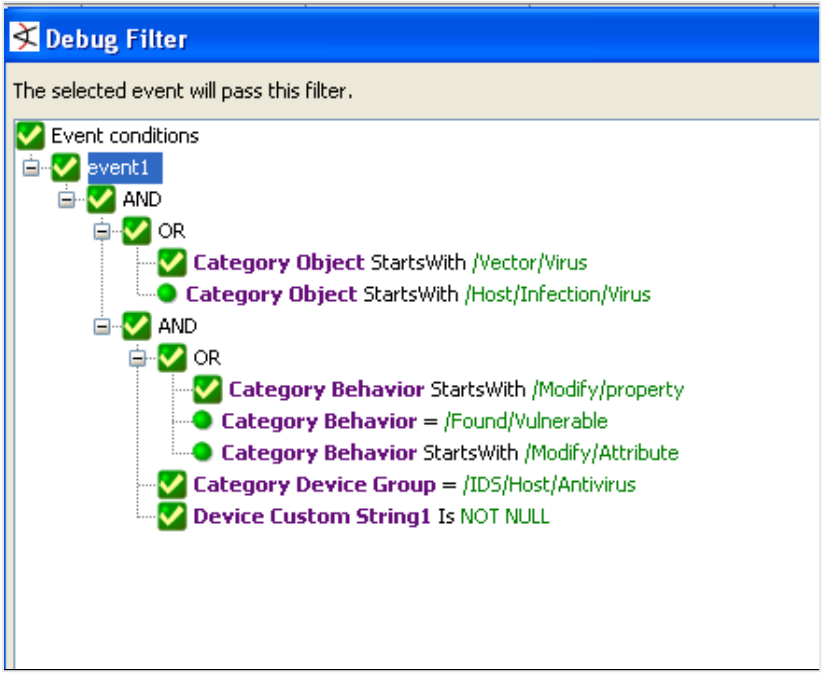
# Filter debugging




Rule should fire with this event now

Fix the filter condition that is failing

Debug filter with the same event



# Rules



# Writing slim ’n fast rules

- Slim rules—don’t consume a lot of memory
- Things to watch out for
  - Partial matches
  - Selectivity of aliases
  - Time window of the rule
  - Grouping

# Join rule: An example

# Partial matches

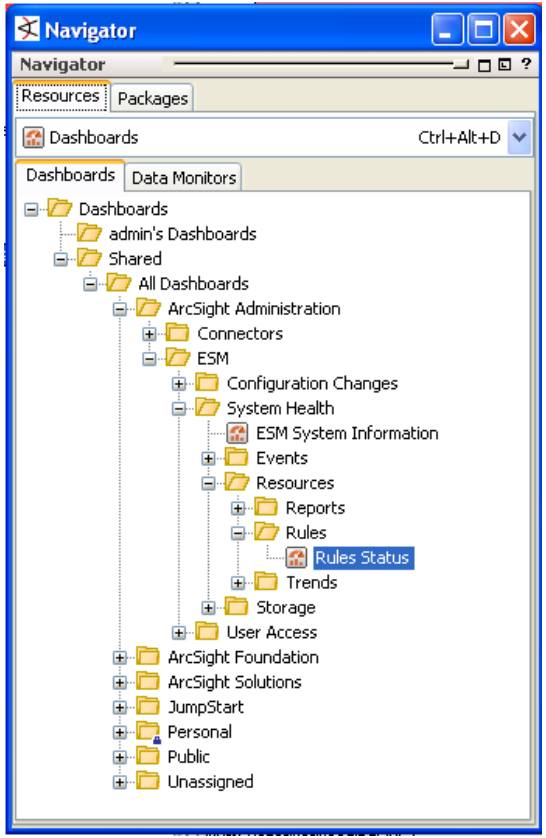
- Partial matches are events that match one of the aliases of a join rule
- These events will be held up in memory waiting for events matching other aliases
- Time window determines how long they will be held in memory

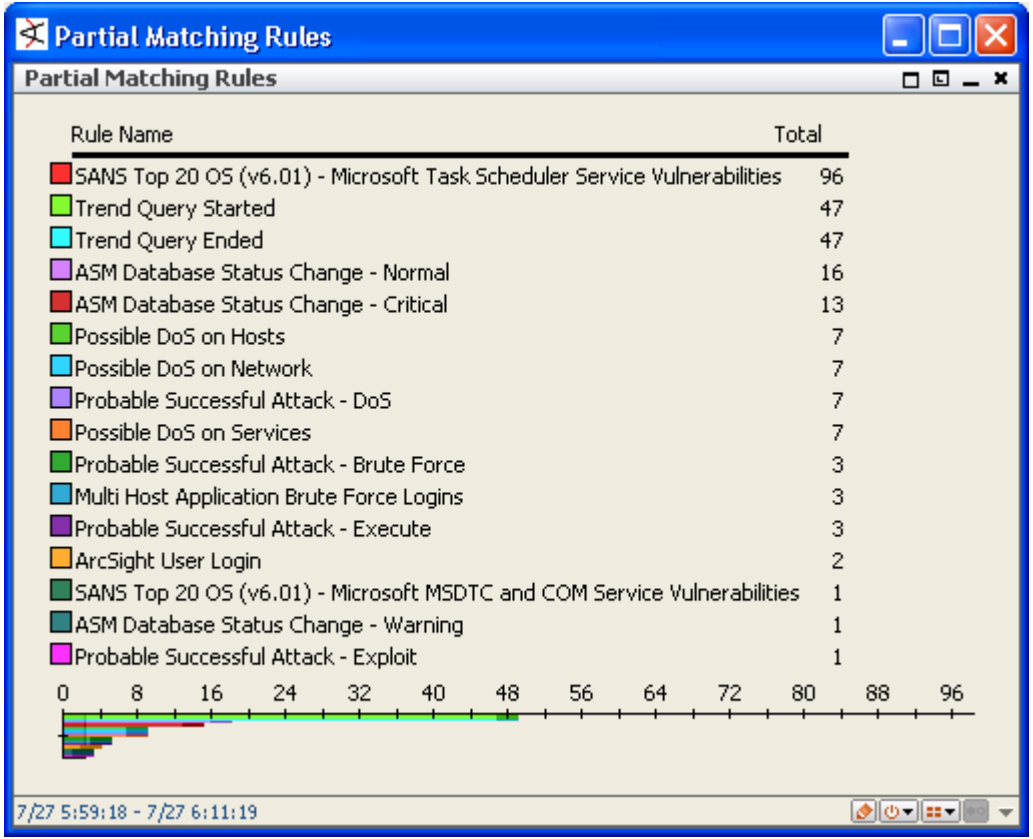
# Making peace with join rules

Make sure that…

- Individual aliases don’t match a lot of events
- Time window is small, typically a few minutes
- Number of aliases are the absolute minimum needed for the use case
- Use Consume After Match Option on each alias whenever possible
  - Use the event only once to fire a rule
  - The event does not have to stay in Rules Engine for the whole time window after it produces a match
  - Use this option to reduce the number of correlation events

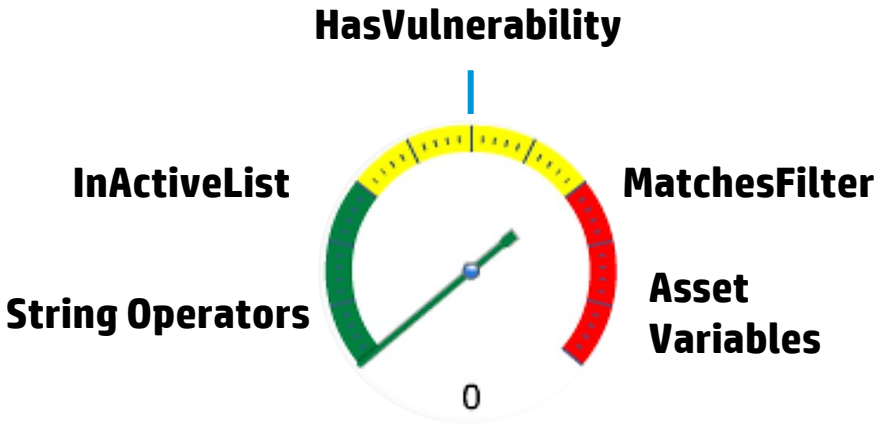
# Looking for partial matches



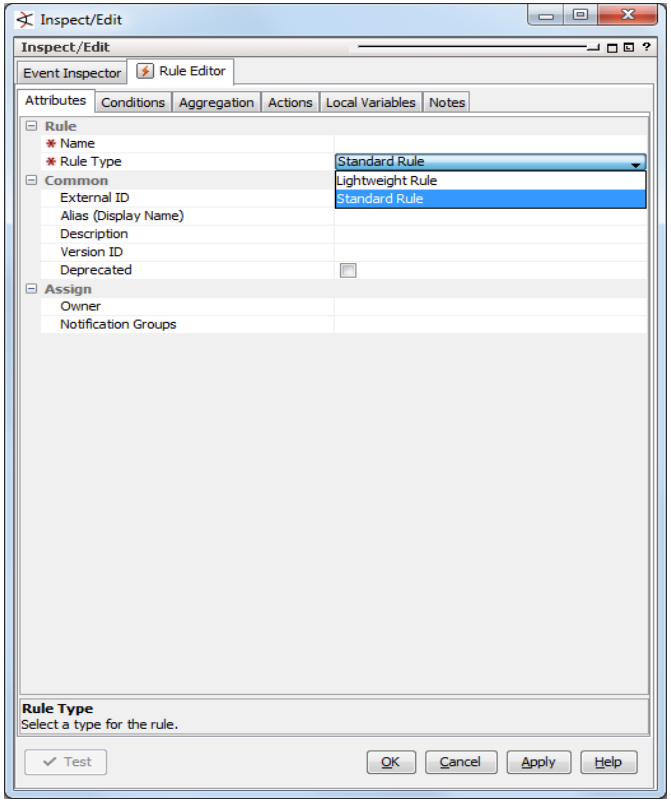
# All operators are not created equal

- Case sensitive string operations are faster than the case insensitive ones
- Some of the costly operators
  - InActiveList—still far better than a join rule with a long time window
  - HasVulnerability—to check if an asset has a particular vulnerability
  - MatchesFilter— can be expensive depending on the filter
  - Conditions on asset based variables
- Multiple conditions in AND/OR put the most costly operator at the end
- Use Global Variables if needed by multiple rules



# Lightweight rules

- New option while creating rules
- Enables a small set of features for faster and simpler rule processing
- Does not generate correlation or audit events (although failures are logged)



# Lightweight rules - when to use

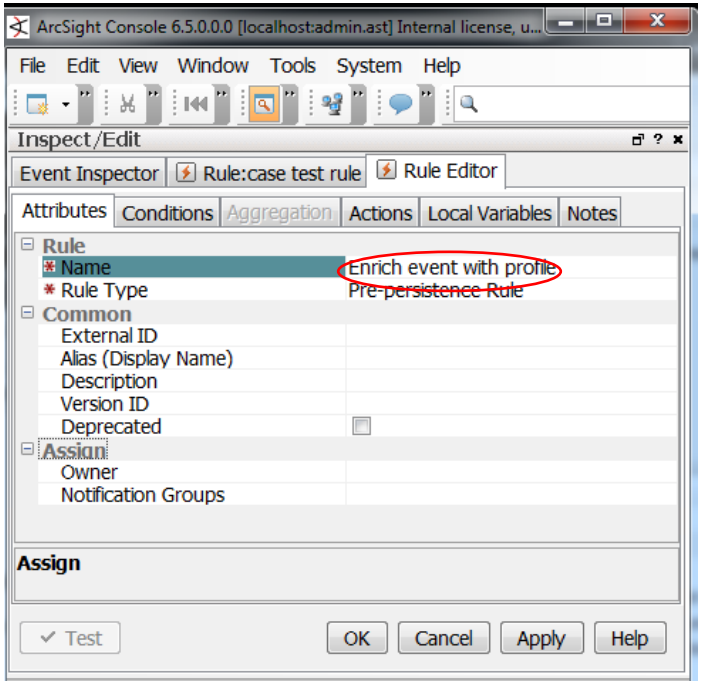
- Use when the rule is used for maintaining data in Active Lists and Session Lists
- Example
  - A rule that maintains the DHCP or VPN Session List by starting and terminating sessions on receiving corresponding events
  - A rule that maintains sum total of transaction amount per user per day in an Active List




# Pre-persistence rules (ESM 5.5)

Designed for event enrichment

- No correlation/audit event, aggregation
- Only action is SetEventField
- SetEventField actions processed prior to event persistence
- Enriched field values available to rules that are evaluated post-persistence



# Pre-persistence rules - when to use

- Use when some calculated information needs to be persisted in events for later use in Reports, Channels etc.
- Example
  - A rule that identifies user based upon information from multiple sources (E.g. DHCP, VPN, Static IP assignments etc.)
  - The user information can be persisted for use in reports later (and avoid costly conditional joins in queries)




# Data monitors

# Resource utilization

- Main resource concerns are CPU and memory
- Non-event data monitors generally just gather and display content… not very heavy
- Event-based data monitors can be heavy
  - Time buckets
  - Number of groups
- Time buckets and number of groups are directly related to the memory consumption

**From light to heavy**

| | |
|---|---|
| **Last state** | #groups |
| **Event graph** | #groups + node graph |
| **Reconciliation** | #groups * 2 |
| **Moving average, statistics, top value** | #groups * #buckets |

# What's a time bucket?

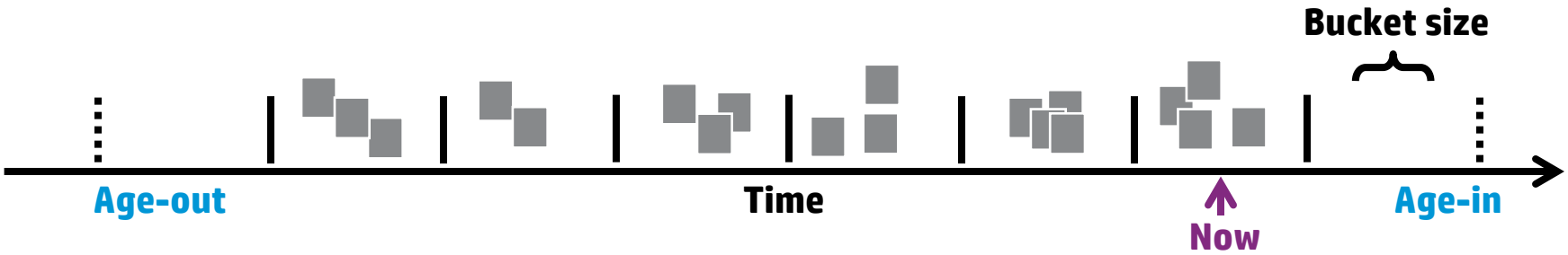
Time buckets group events by time

Time buckets are used for aging out the data

Example: Bucket size = 300 (in seconds)

# of buckets = 12

This means: (12 time buckets) * (5 minutes/bucket) = 1 hour of data



# Choosing time buckets

- Hard to estimate
- Tip: find out the time range for the data in which you are interested
- Choose bucket size to get enough data to be statistically significant

  Example:

  Calculating moving average over the last 1 hour

~~3600 time buckets~~

~~Bucket size = 1 second~~
~~#of buckets = 3600~~

**12 time buckets**

Bucket size = 300 seconds
#of buckets = 12

~~1 time bucket~~

~~Bucket size = 3600 seconds~~
~~#of buckets = 1~~

# Group by

Number of groups adds to memory consumption

- Efficient grouping: event name, (address + port)
- Inefficient grouping: event ID, time, bytes in

# Data monitor memory usage

## CapsManager

- https://hostname:8443/arcsight/web/manage.jsp
- Click on CapsManager and scroll way down to Arcsight:service=CapsManager, id=Datamonitor Caps Manager

| Consumer Name | Priority | Delta (KB) | Estimated Usage (KB) | UsageInfo | Past 4 Usages (KB) |
|---|---|---|---|---|---|
| EventGraph: ArcSight Events Live | 3 | 0 | 2040 | 2000 events | 2040, 2040, 2040, 2040 |
| LastNEvent: Worm Activity Status | 3 | 0 | 31 | 30 events | 31, 31, 31, 31 |
| BucketizedTopValueCountsEvent: Top Agents | 3 | 0 | 16 | 54 counts, fields [Agent Name, Agent ID] | 16, 16, 16, 16 |
| LastNEvent: Last Failed Logins | 3 | 0 | 15 | 15 events | 15, 15, 15, 15 |
| LastNEvent: Last Attacks | 3 | 0 | 15 | 15 events | 15, 15, 15, 15 |
| LastNEvent: Password Change | 3 | 0 | 15 | 15 events | 15, 15, 15, 15 |
| LastNEvent: Recent Events | 3 | 0 | 15 | 15 events | 15, 15, 15, 15 |
| LastNEvent: Recent Fired Rules | 3 | 0 | 15 | 15 events | 15, 15, 15, 15 |
| MovingAverageEvent: ASM Load Overview - Last 24 hours | 3 | 0 | 12 | 23 cells, 24 samples/cell, aggregate fields [Device Event Category] | 12, 12, 12, 12 |
| Asset Group Count: Business Impact by Role - Successful Attacks | 3 | 0 | 11 | 27 device groups | 11, 11, 11, 11 |
| Rest (144 more) | - | 0 | 463 | - | 463, 463, 463, 463 |
| Total | - | 0 | 2648 | - | 2648, 2648, 2648, 2648 |

# Always remember

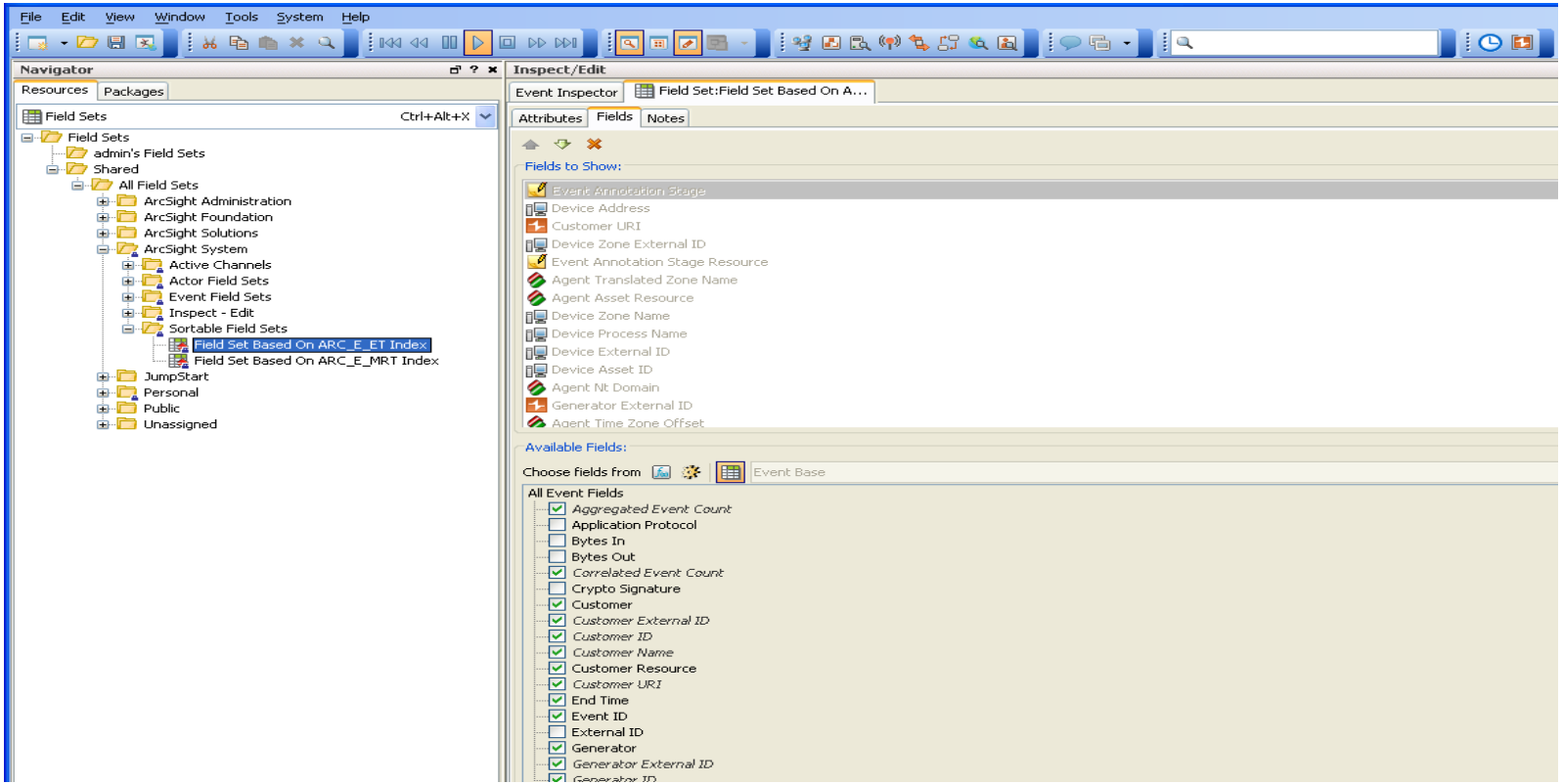
- Choose the right type of data monitor
- Filter
  - Restrict events as much as possible
  - Data monitors process each event passed by the filter
- Have only those data monitors enabled that you need for better performance
- A single poorly configured data monitor can degrade manager performance
- Restrict who can edit data monitors
  - Use data monitor deploy permission

# Reports

# Reports

- Query on indexed columns (Oracle)
- Query on small time ranges
  - Querying on long time ranges for a value that's not indexed is going to be slooooooooooooooow
- Partial list of indexed columns
  - End time
  - Manager receipt time
  - Source/destination address, port
  - Event type
  - Originator
  - Customer
  - Type/priority/generator
- All columns are indexed on CORRE

# Which fields are indexed?



# Performance tips and tricks

- Query on the ID field instead URI
  - URI is derived from ID in almost all cases
- Watch out for variables
  - Asset-based variables are heavier than time-based variables
- Keep string comparisons case sensitive
  - Indexes are useless for case insensitive string operation
- Query on end time instead of manager receipt time
  - Events are partitioned by end time, hence Oracle would know exactly which partition to scan

# Reporting on report performance

- Audit events are generated whenever a report is run
- You can
  - Find out the longest running reports
  - How many reports are being run per day
  - Notify when a report run takes longer than say 30 minutes

# Trend reports

# Trend reports

- Similar to scheduled queries
- Results of queries stored in database for further reporting and querying
- Results can be persisted for much longer than the events
  - Especially useful if the result of a query is much smaller than the events processed by the query
- Much of the data and table management is done by the system




# Trend report: Example

- Goal: print a set of 20 reports at the end of every month
- Example report
  - Daily counts of events blocked by firewall in last month
- Problem
  - Report takes few hours
  - Enormous amount of data to be scanned for each report
  - Re-running the report adds another few hours

# Trend report: Example

- All the reports can be evaluated incrementally using trends
- Monthly query can be broken in several smaller daily queries (daily trend)
- The data will be stored in a trend table on daily basis
  - Daily counts of events blocked by firewall in last day
- At the end of the month, the report can run on this daily trend

# Event annotation performance enhancement

## Problem: Slow queries when large volume of event annotation data – MySQL

Turn on event.annotation.optimization.enabled to true, if false

- Default value is true

Dynamic optimization using temporary table

Approximately 12X-18X improvement from ESM 6.0c

# MySQL sorting performance enhancement

## Problem: Excessive temporary file space used when sorting event data – MySQL

Use only the portion of event field that is required

- Use global/local variable and ArcSight SUBSTRING on Event field

# Summary

**There are many technical ways to achieve the same business goal**

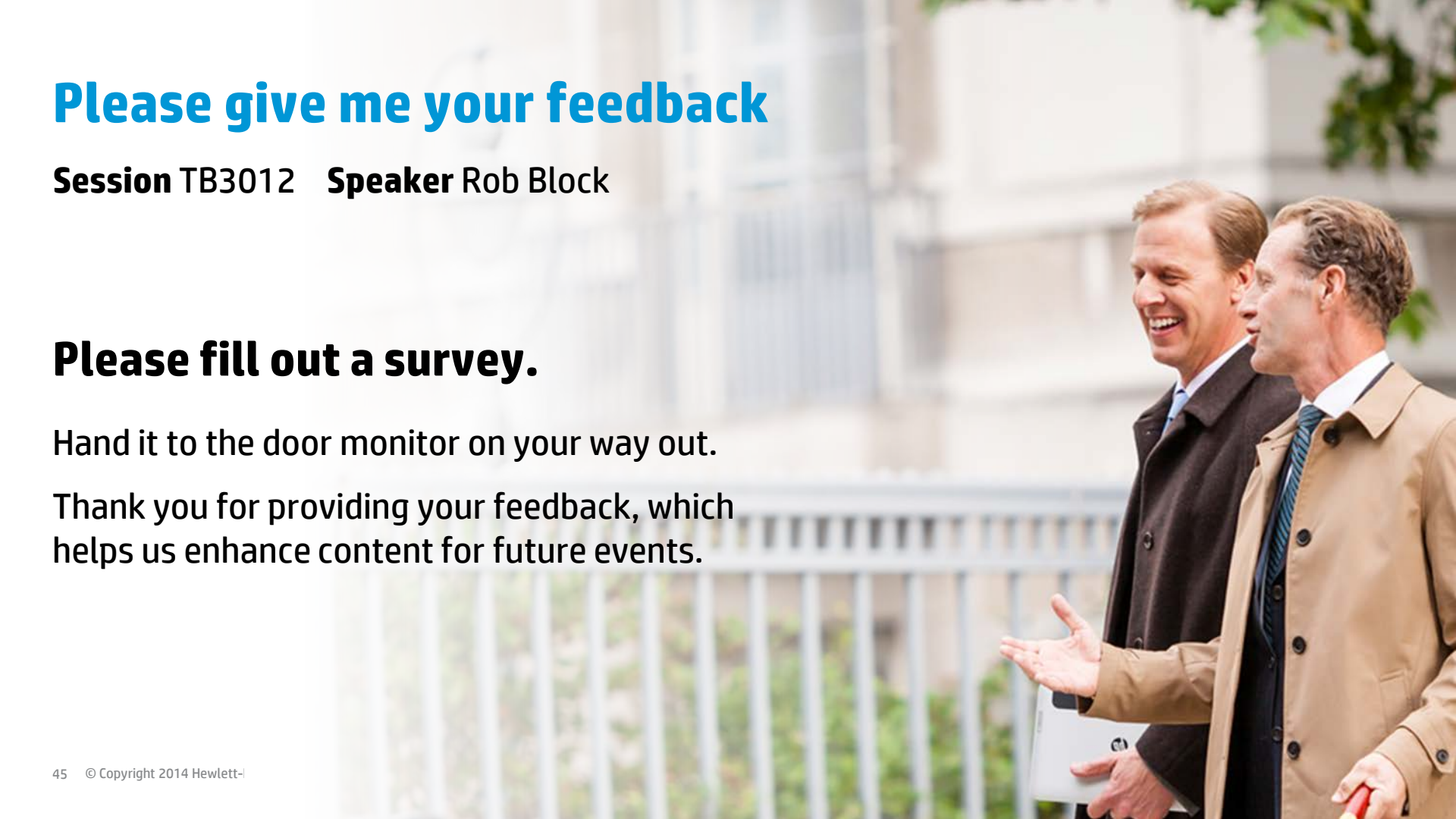
**Every piece of content you write has a performance impact—make an educated choice**

# Questions?



# Please give me your feedback

**Session** TB3012 **Speaker** Rob Block



## Please fill out a survey.

Hand it to the door monitor on your way out.

Thank you for providing your feedback, which helps us enhance content for future events.

# Thank you



Make it matter.