General Rules:

1. Identify what information the app will bring to the users? Is the information useful for troubleshooting, or monitoring?
2. Each app should have a landing dashboard, usually called an overview dashboard that provides a high level view of the health/status of the system/product the app is for. From there, users can get more specific details in other dashboards.
3. For dashboards, limit the number of panels to 9 or fewer. Dashboards with higher number of panels can load slowly.
4. For logs, identify a set of **core** fields that you will parse out **for all the queries**, even though you **might aggregate on just a few fields in the end**. Some examples of common fields are user name, source/target host, application name. These fields will be very handy when new users drill-down from an aggregated panel to the raw messages (via the Messages tab), they will see more details instead of the raw messages. Furthermore, you can these fields as dashboard filters across your dashboards.
5. **Searches and Dashboard, Panel Titles Naming Convention.** Keywords and Names should be capitalized in the first letter, but not articles like a, an, the. For example: "Events by Hour", "Events Overtime". So basically, use title caps, not sentence caps.
6. Use search templates when you can - they can later be used as filters on dashboards (although for dashboards they don't offer any extra value - their main value is for non-Sumo savvy users when they drill-down into the search UI).

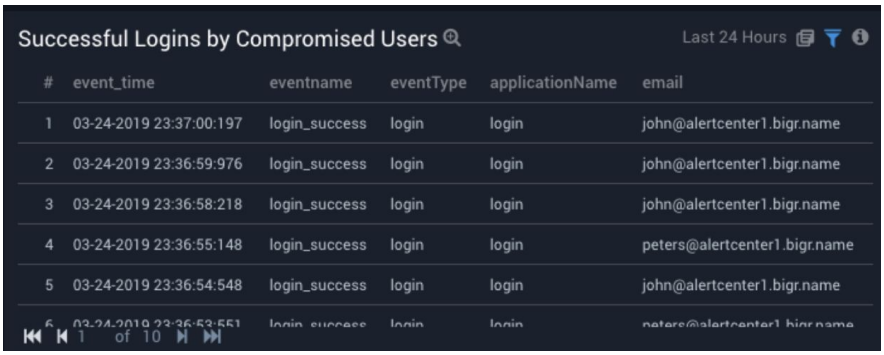
Dashboards:

1. **Alignment.** Make sure your dashboard panels align nicely on the edges. Also try to fill the Dashboards as much as you can.
2. **Filters.** Use filters in your Dashboards. Try to make them match all Panels if possible. Also: to find which panels a filter applies to, hover the mouse over the filter icon, and check the highlighted panels.
3. **Overview Dashboard.** An Overview Dashboard is required (see Rule 2 above), with the naming convention: **"App Name - Overview". Yes, a space, a dash, followed by a space.** One should link other dashboards with this dashboard to provide a seamless flow. Refer the docs on how to link dashboard.
4. **Time Ranges.** Each Dashboard should have at most two time ranges for the Panels. One for short-term, one for long-term data.
5. Dark theme dashboards are usually preferred over white theme

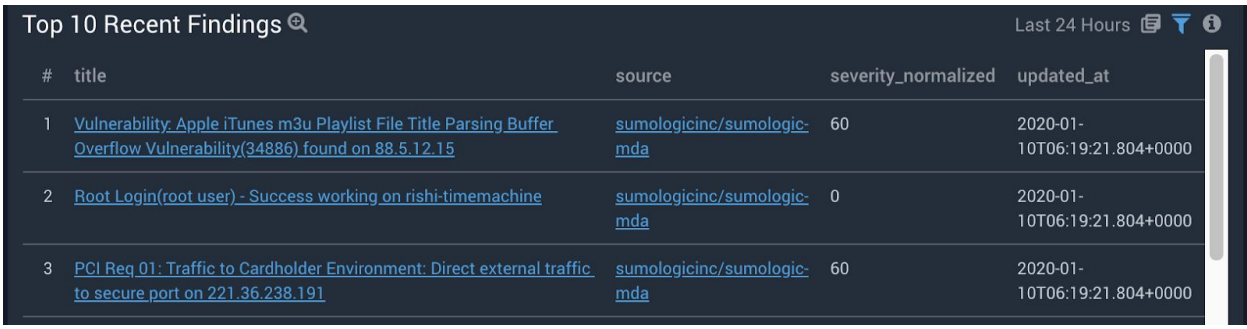
Panels

1. For Pie chart always sort the list by count.

2. When showing data in tables
    a. Users may not be interested in looking into Informational and Debug Log Levels. Show the Higher severity information (Emergency/ Critical, Error etc) in separate panels and all other combined in another panel
    b. The order of columns should be such that the timestamp field should come first and then important fields like email etc.

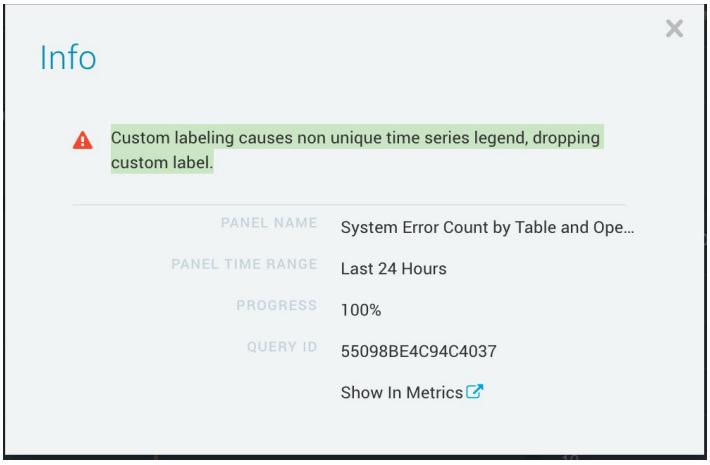


    c. Columns names should be snakecase
    d. Try to make the panel interactive by creating dynamic links like the example below



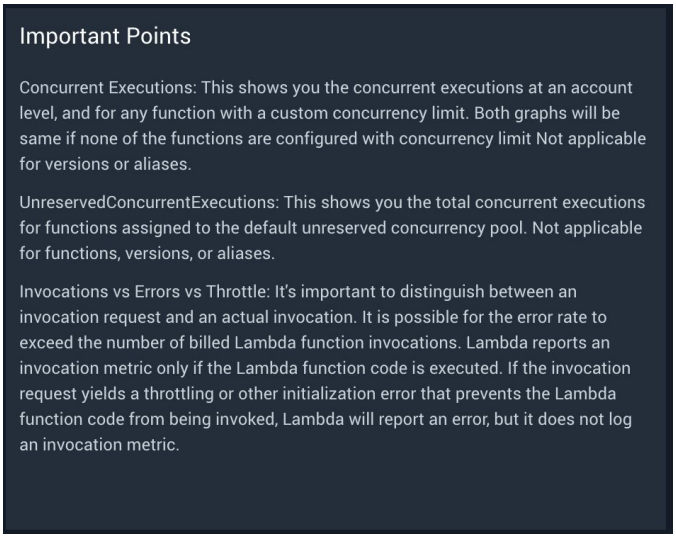
    Refer the docs to see examples on how to use tourl operator.

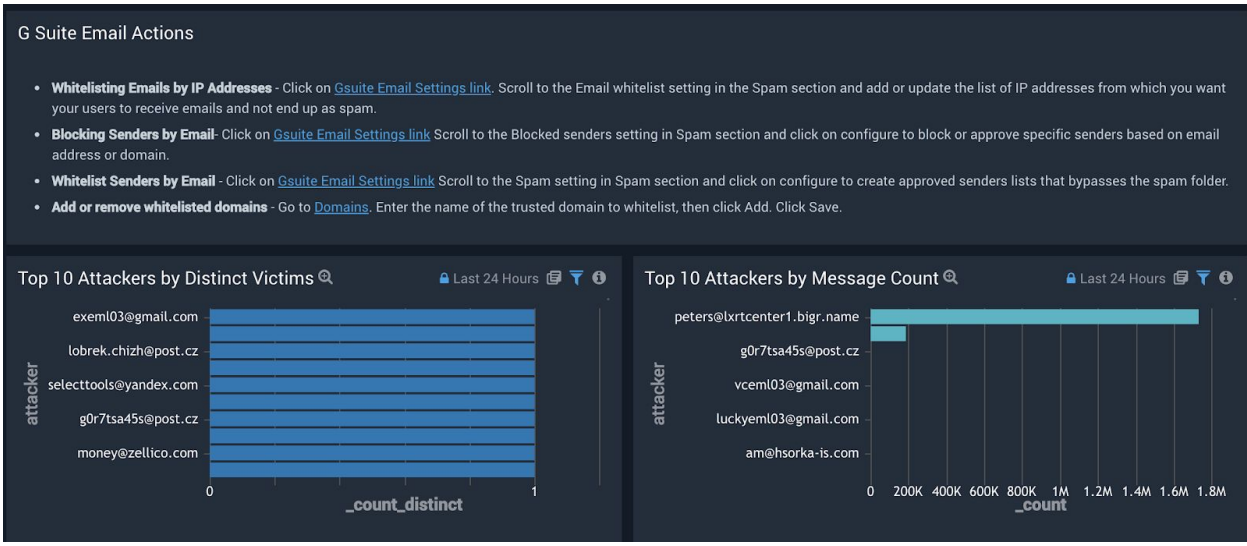
3. General Feedback for all metrics panels - Use custom labels for metrics.Do not use camel case labels use snake case instead - e.g. change uniqueContainers to unique_containers. Make sure that custom labelling is not dropped due to “Custom labeling causes non unique time series legend, dropping custom label”.



4. For filters if you use the same field name in logs (say for region), the filter can be shared for both logs and metrics.

5. Use Text Panels extensively to
    a. To describe important fields for better understanding of the information shown in the dashboards.
    b. To give links to action items which utilize information from the panels.





6. For Single Value Panels show one should change background color depending on if there is any data. For example say if there is a panel showing Total Errors then one should make it green if there are no errors else it should be red.
   Green (#008000), Orange (#FFA500), Yellow (#ffc508), Red (#FF0000)

Search Queries [VERY IMPORTANT]

7. Use **_sourceCategory** as a metadata tag for your input data source for queries. Please DON'T use
   _source, _collector metadata tags in your queries.

   Example :
   Below query :

```
_source="kdata" and _collector="KCollector" and
_sourceCategory="Kdetections"
| json "category","etype", "eid" as category, etype, eid
| where etype="principal"
| count by eid
```

   Should be :

```
_sourceCategory="detections"
| json "category","etype", "eid" as category, etype, eid
| where etype="principal"
| count by eid
```

   More info on _sourceCategory here

8. By default keywords in query scope use AND operator

   _sourceCategory=Labs/SecDemo/vpcflow OR _sourceCategory=vpc_logs_single ACCEPT

   (_sourceCategory=Labs/SecDemo/vpcflow OR _sourceCategory=vpc_logs_single) ACCEPT

   Both of the above queries have different scope.
9. Always put aggregates first before lookups

   _sourceCategory=Labs/barracuda " TR "
   | parse regex "(?<Unit_Name>[^]+) TR(?<Log>.*)"
   | split Log delim=' ' extract 4 as Client_Ip, 2 as Service_Ip, 3 as Service_Port
   | timeslice 30m
   | count by _timeslice, Client_Ip
   | lookup latitude, longitude, country_code, country_name, region, city, postal_code, area_code, metro_code from geo://default on ip = Client_Ip

10. Do not use heavy operators or joins with large time ranges.
11. **Don't Use Custom Lookups.** Don't use any custom **lookup from a local file.** If you feel the lookup information is static, host it on a static location (e.g AWS S3 bucket) and use Https lookup. We provide static geolookup , port numbers, protocol numbers, and emoji that you can use.

    Refer the docs to see examples on how to use lookup operator.

12. Always try to improve query performance by adding keywords to scope.

    _sourceCategory=Labs/barracuda **“AUDIT" "UNSUCCESSFUL_LOGIN"**| parse regex "(?<Activity_Time>\d\d\d\d-\d\d-\d\d\s{1,3}.*)(?<Unit_Name>[^]+) AUDIT(?<Log>.*)"| split Log delim=' ' extract 2 as Admin_Name, 3 as Client_Type, 4 as Login_Ip, 5 as Login_Port, 6 as Transaction_Type, 7 as Transaction_Id, 8 as Command_Name, 9 as Change_Type, 10 as Object_Type, 11 as Object_Name, 12 as Variable_Name, 13 as Old_Value, 14 as New_Value, 15 as Additional_Data| where Transaction_Type matches "UNSUCCESSFUL_LOGIN"| timeslice 15m|count as Count by _timeslice| fillmissing timeslice(15m)

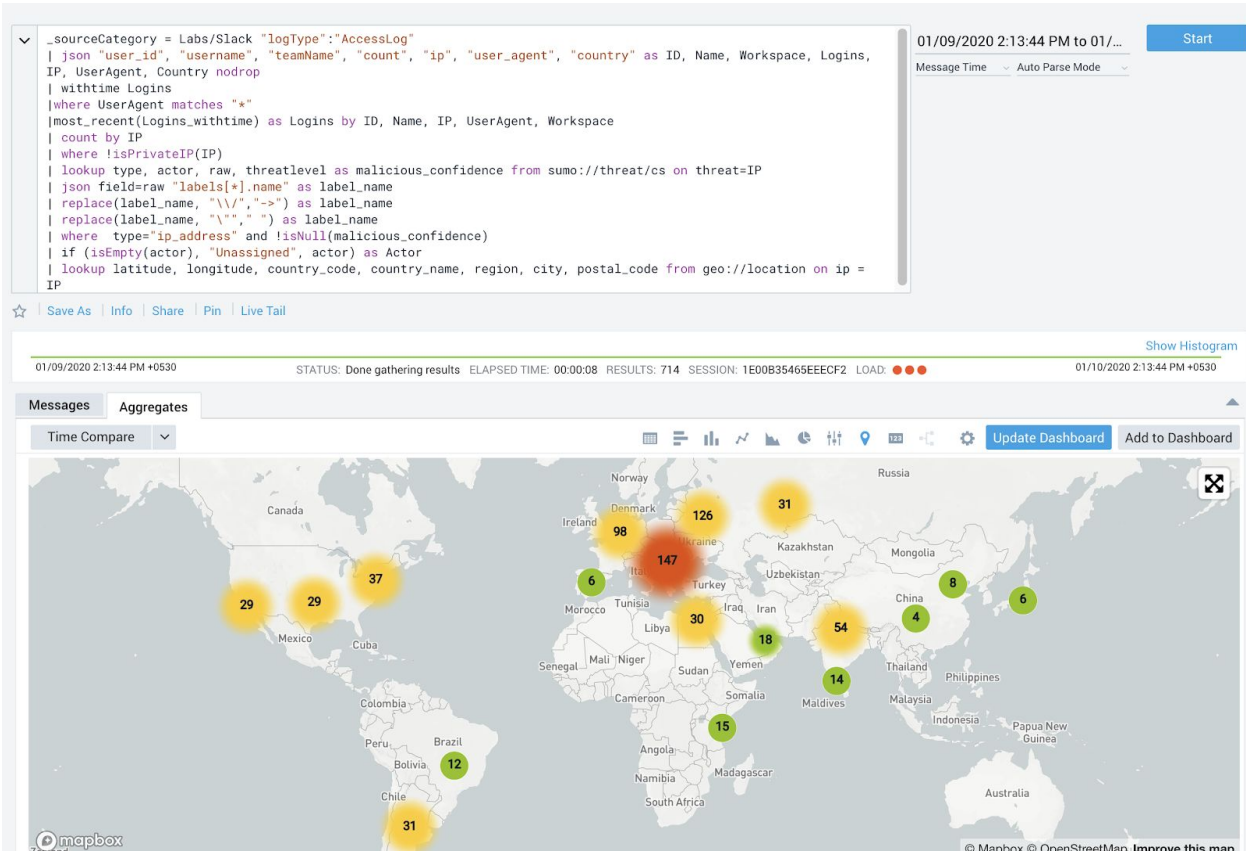
13. Panels where predict with ar model is used. It requires minimum 100 points for proper predictions else it gives warning. In 60 min, 1m timeslice will have approx 60 points. So good way is to have 30s timeslice or increase timerange to 3 hrs.

14. Be careful with parse, json they also filter the logs which do not have the matched fields use nodrop for such type of logs.

15. JSON auto extracts up to 100 fields; additional fields must be explicitly keyed. Do not use json auto if you want only a few fields.

(_sourceCategory="securityhub_findings" OR _sourceCategory="Labs/AWS/SecurityHub")
| **json "AwsAccountId", "Id", "GeneratorId", "ProductArn", "CreatedAt", "UpdatedAt", "Resources", "Severity.Normalized", "SourceUrl", "Title","Types", "Compliance.Status" as aws_account_id, finding_id, generator_id, product_arn, created_at, updated_at, resources, severity_normalized, sourceurl, raw_title, finding_types, compliance_status nodrop**
| topk(1, updated_at) by finding_id

16. While dealing with IP addresses, leverage operators like isPrivate, isPublicIP,isValidIP to optimize the query performance.



17. Provide unit of size (KB) somewhere (Variable Name or some GUI component) when dealing with fields with units.

_sourceCategory=Labs/barracuda " TR "| parse regex "(?<Unit_Name>[^]+) TR(?<Log>.*)"| split Log delim=' ' extract 13 as **Bytes_Sent**, 4 as Client_Ip, 2 as Service_Ip, 3 as Service_Port, 21 as Response_Type, 17 as Backend_Server| timeslice 15m| **round((Bytes_Sent / 1024),2) as Bytes_Sent| sum(Bytes_Sent) as Total_Response_Size_KB** by _timeslice| fillmissing timeslice(15m). You can also use size suffix as mentioned in docs.