Forward-looking statements

This presentation may contain forward-looking statements regarding future events, plans or the expected financial performance of our company, including our expectations regarding our products, technology, strategy, customers, markets, acquisitions and investments. These statements reflect management's current expectations, estimates and assumptions based on the information currently available to us. These forward-looking statements are not guarantees of future performance and involve significant risks, uncertainties and other factors that may cause our actual results, performance or achievements to be materially different from results, performance or achievements expressed or implied by the forward-looking statements contained in this presentation.

For additional information about factors that could cause actual results to differ materially from those described in the forward-looking statements made in this presentation, please refer to our periodic reports and other filings with the SEC, including the risk factors identified in our most recent quarterly reports on Form 10-Q and annual reports on Form 10-K, copies of which may be obtained by visiting the Splunk Investor Relations website at www.investors.splunk.com or the SEC's website at www.sec.gov. The forward-looking statements made in this presentation are made as of the time and date of this presentation. If reviewed after the initial presentation, even if made available by us, on our website or otherwise, it may not contain current or accurate information. We disclaim any obligation to update or revise any forward-looking statement based on new information, future events or otherwise, except as required by applicable law.

In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. We undertake no obligation either to develop the features or functionalities described, in beta or in preview (used interchangeably), or to include any such feature or functionality in a future release.

splunk> .conf24

Maximizing Splunk Core: Analyzing Splunk Searches Using audittrail and Native Splunk Telemetry

PLA1837B

Who are you?

Architect

Engineer

Admin

Manager

Developer

Someone who wants to...

- Identify what data sources are powering your searches
- Gain more value from your Splunk environment
- Become familiar with OOTB Splunk Default logging
- Nerd out on some neat SPL & Splunk Quirks

Base Knowledge Pre-Reqs

to get the most out of this presentation...

- Familiar with what a Splunk Search is
- You've seen some internal data sources

What are you gaining?

How to get the data for insights into your environment!
Like "What data sources are my searches using?"

index_reference	dc_searches
wineventlog	18
panlogs	12
firewall	12

label				index_keywords	sourcetype_keywords
Sample Data Generation - Add NOT	1717122660_34555	search index=_internal OR index=_telemetry NOT sourcetype=splunkd_ui_access NOT sourcetype="*modular_input*" NOT index=_introspection | stats count by index | append [search index=_audit sourcetype=audittrail | stats count by index]	index::_audit sourcetype::audittrail index::_internal index::_telemetry	_audit _internal _telemetry	audittrail

https://github.com/TheWoodRanger/presentation-conf_24_audittrail_native_telemetry

Objectives

It's our objective for you to leave this talk with...

Basics

A basic familiarity of the default Splunk data sources available in your environment.

Details

The knowledge of how to get the Native Splunk Data to answer that key question -

"What data sources are my searches using?"

R&R

References & Resources that are available to dive further & add value.

Agenda

Introduction & Level Set

- Introductions
  - Objectives
  - Background
- Framing the scope of this presentation
- Overview of Default Indexes

Getting Technical

- Identifying Problematic Searches
- Unifying Default Indexes for insight
- Data Source Usage using:
  - Instrumentation
  - Default Indexes
  - REST
- Next Steps & Resources
  - Why reinvent the wheel?

Who are these guys?

Ryan Wood

Sr. Splunk Solutions Architect
GuidePoint Security

Rich Galloway

Technical Account Manager
Splunk

Rich Galloway

Background

- Splunk user since 2012
- SplunkTrust member since 2016
- Professional Services provider for 6 years
- Co-leader of the DC user group

"Wherever you go, there you are"

Ryan Wood

Background

- Certifications:
  - Splunk Core Certified Consultant
  - Splunk Enterprise Security Certified Admin
  - Security+
- Daily Splunk User for over 11 years
- Delivering Splunk Professional Services for over 6 years
- Conf 2023 Speaker
  - *Maximizing Splunk SPL™: Foreach and the Power of Iterative, Templatized Evals*
- Passionate about enabling others to find success

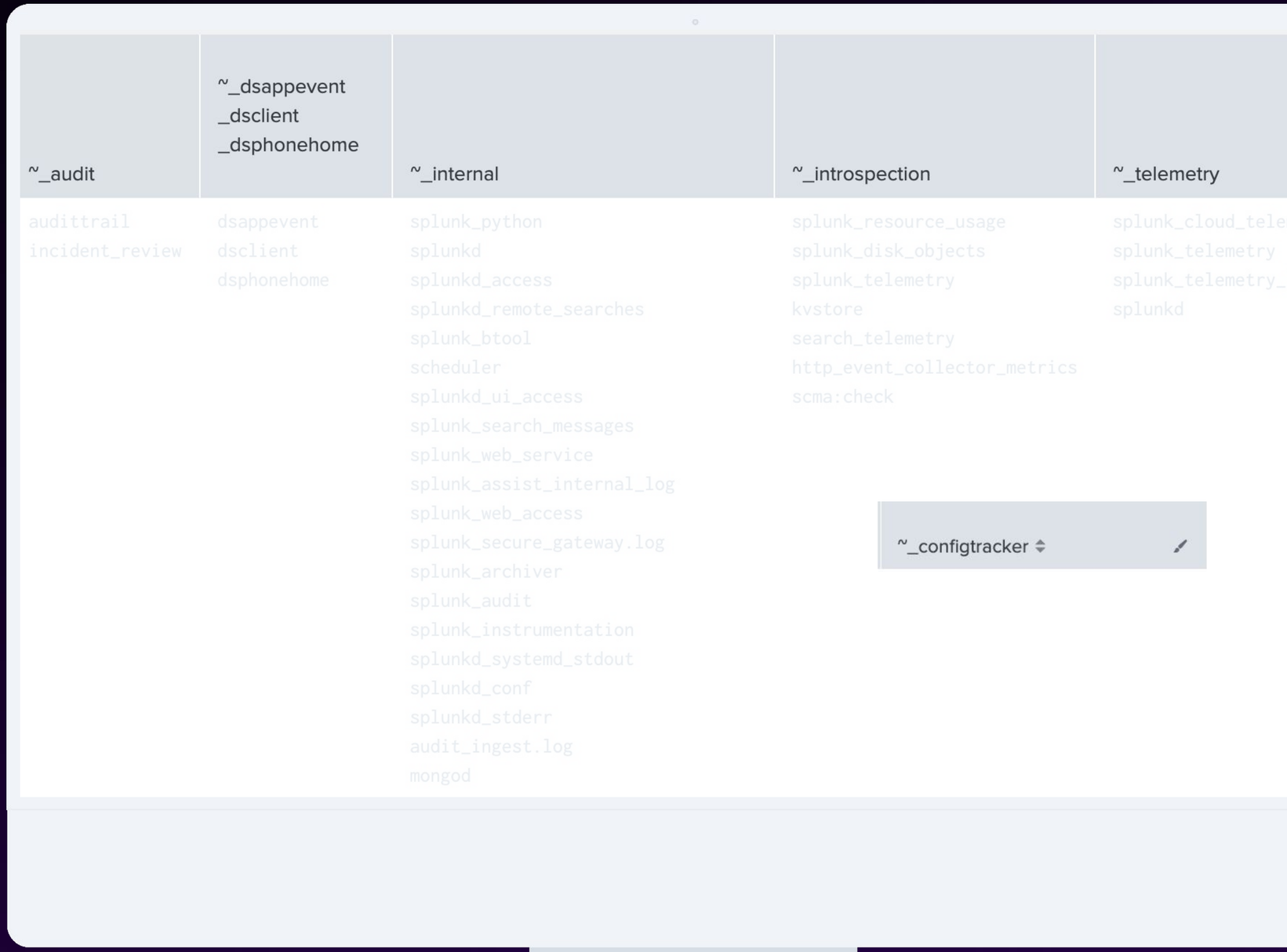
"The only time I don't enjoy my job is when I'm doing repetitive, tedious things."

Splunk Default Indexes

Framing the Scope of This Presentation

There's a *Lot* of Default Data Sources

- Splunk does a lot, so there's a lot of logging about what it's doing.
- 8 underscore indexes by default as of v9.2

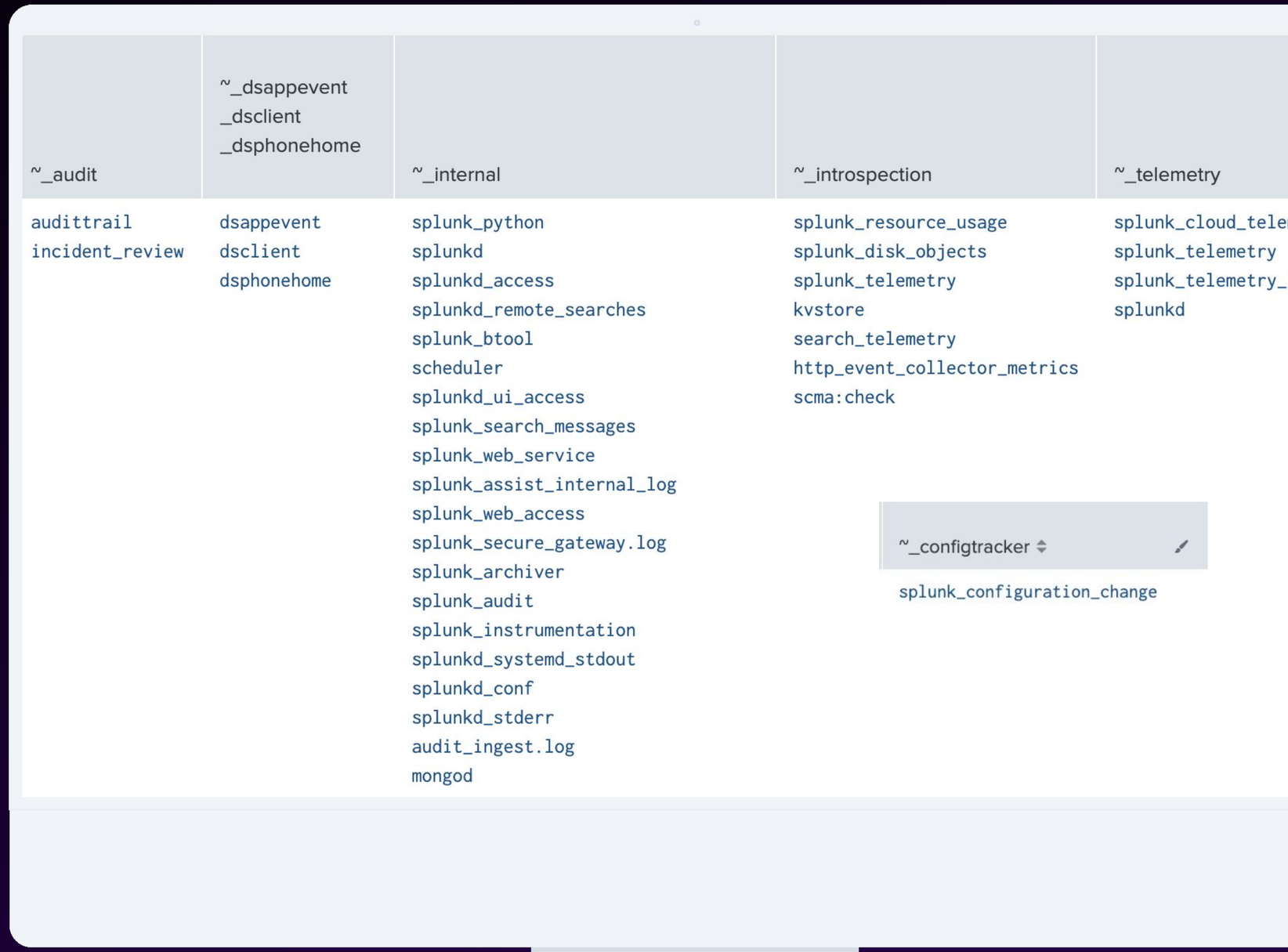


~_audit	~_dsappevent _dsclient _dsphonehome	~_internal	~_introspection	~_telemetry
audittrail	dsappevent	splunk_python	splunk_resource_usage	splunk_cloud_tele
incident_review	dsclient	splunkd	splunk_disk_objects	splunk_telemetry
	dsphonehome	splunkd_access	splunk_telemetry	splunk_telemetry_
		splunkd_remote_searches	kvstore	splunkd
		splunk_btool	search_telemetry	
		scheduler	http_event_collector_metrics	
		splunkd_ui_access	scma:check	
		splunk_search_messages		
		splunk_web_service		
		splunk_assist_internal_log		
		splunk_web_access		
		splunk_secure_gateway.log		
		splunk_archiver		
		splunk_audit		
		splunk_instrumentation		
		splunkd_systemd_stdout		
		splunkd_conf		
		splunkd_stderr		
		audit_ingest.log		
		mongod		

~_configtracker

There's a *Lot* of Default Data Sources

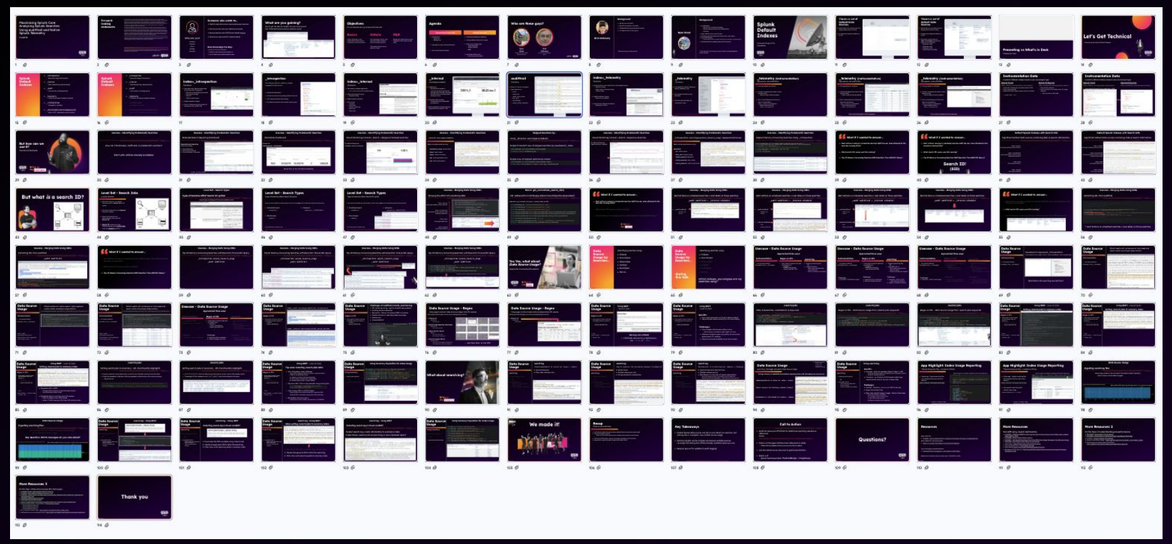
- Splunk does a lot, so there's a lot of logging about what it's doing.
- 8 underscore indexes by default as of v9.2
- 144 unique combinations of **index + sourcetype** found in our reference environments:
  - Enterprise v9.0.4
  - Enterprise v9.1.2
  - Enterprise v9.2.0
  - Cloud Victoria v9.1.2308

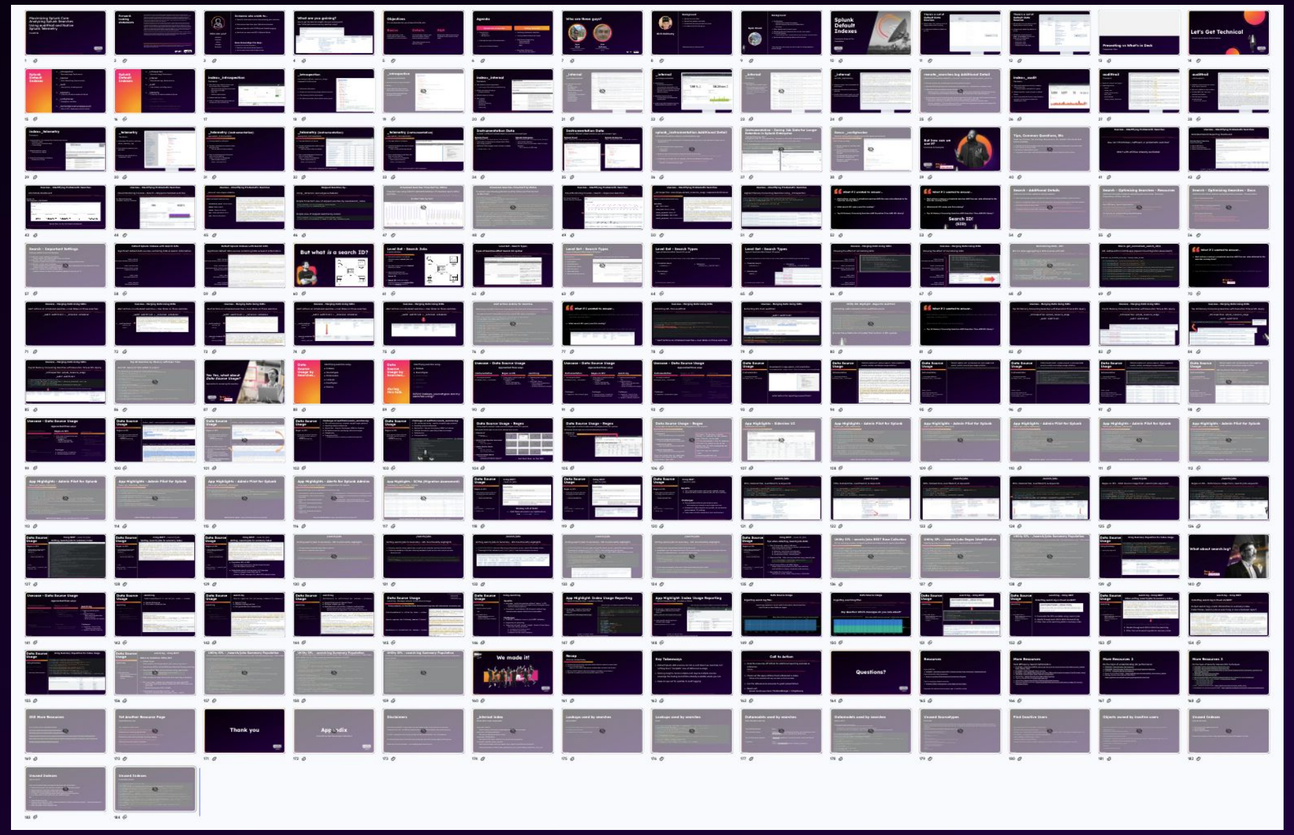


~_audit	~_dsappevent _dsclient _dsphonehome	~_internal	~_introspection	~_telemetry
audittrail	dsappevent	splunk_python	splunk_resource_usage	splunk_cloud_tele
incident_review	dsclient	splunkd	splunk_disk_objects	splunk_telemetry
	dsphonehome	splunkd_access	splunk_telemetry	splunk_telemetry_
		splunkd_remote_searches	kvstore	splunkd
		splunk_btool	search_telemetry	
		scheduler	http_event_collector_metrics	
		splunkd_ui_access	scma:check	
		splunk_search_messages		
		splunk_web_service		
		splunk_assist_internal_log		
		splunk_web_access		
		splunk_secure_gateway.log		
		splunk_archiver		
		splunk_audit		
		splunk_instrumentation		
		splunkd_systemd_stdout		
		splunkd_conf		
		splunkd_stderr		
		audit_ingest.log		
		mongod		

~_configtracker

splunk_configuration_change





Presenting vs What's in Deck

Comparison View

Let's Get Technical

Introducing the Splunk Default Indexes

Splunk Default Indexes

- _introspection
  - Resource Usage, Performance
- _internal
  - Most internal logs, Splunk activity
- _audit
  - User activity, including search
- _telemetry
  - Collection of metrics related to Splunk
- _configtracker
  - Changes to .conf files
- _ds(client|phonehome|appevent)
  - New in 9.2.0 - deployment server activity

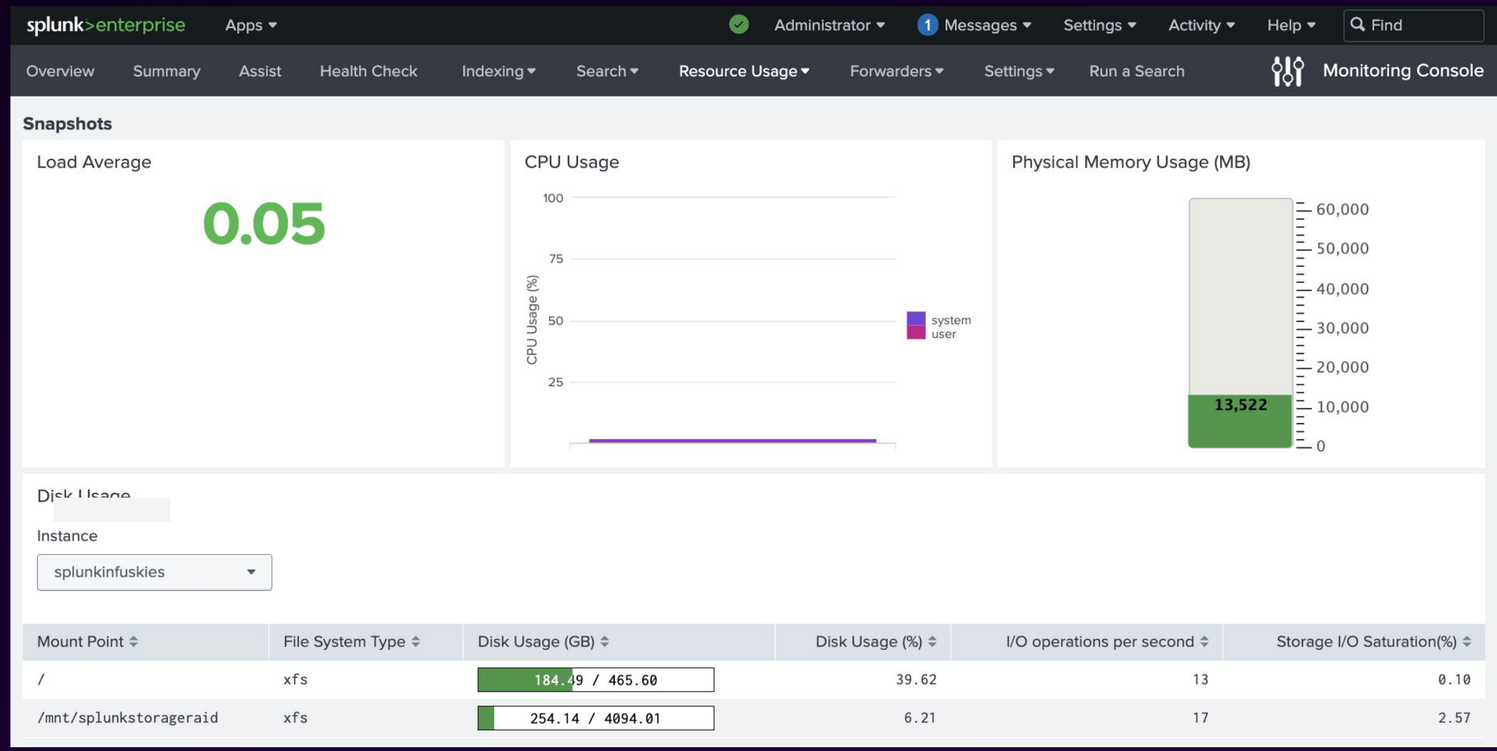
Splunk Default Indexes

- _introspection
  - Resource Usage, Performance
- _internal
  - Most internal logs, Splunk activity
- _audit
  - User activity, including search
- _telemetry
  - Collection of metrics related to Splunk
- _configtracker
  - Changes to .conf files
- _ds(client|phonehome|appevent)
  - New in 9.2.0 - deployment server activity

index=_introspection

The basics

- Data about your Splunk instance and environment. 20+ components.
  - Resource usage by Splunk processes
  - Host resource usage
  - Disk I/O
  - KVStore performance
- Default retention: 14 days
- Data is collected every 10 seconds (10 minutes on UFs) via REST call

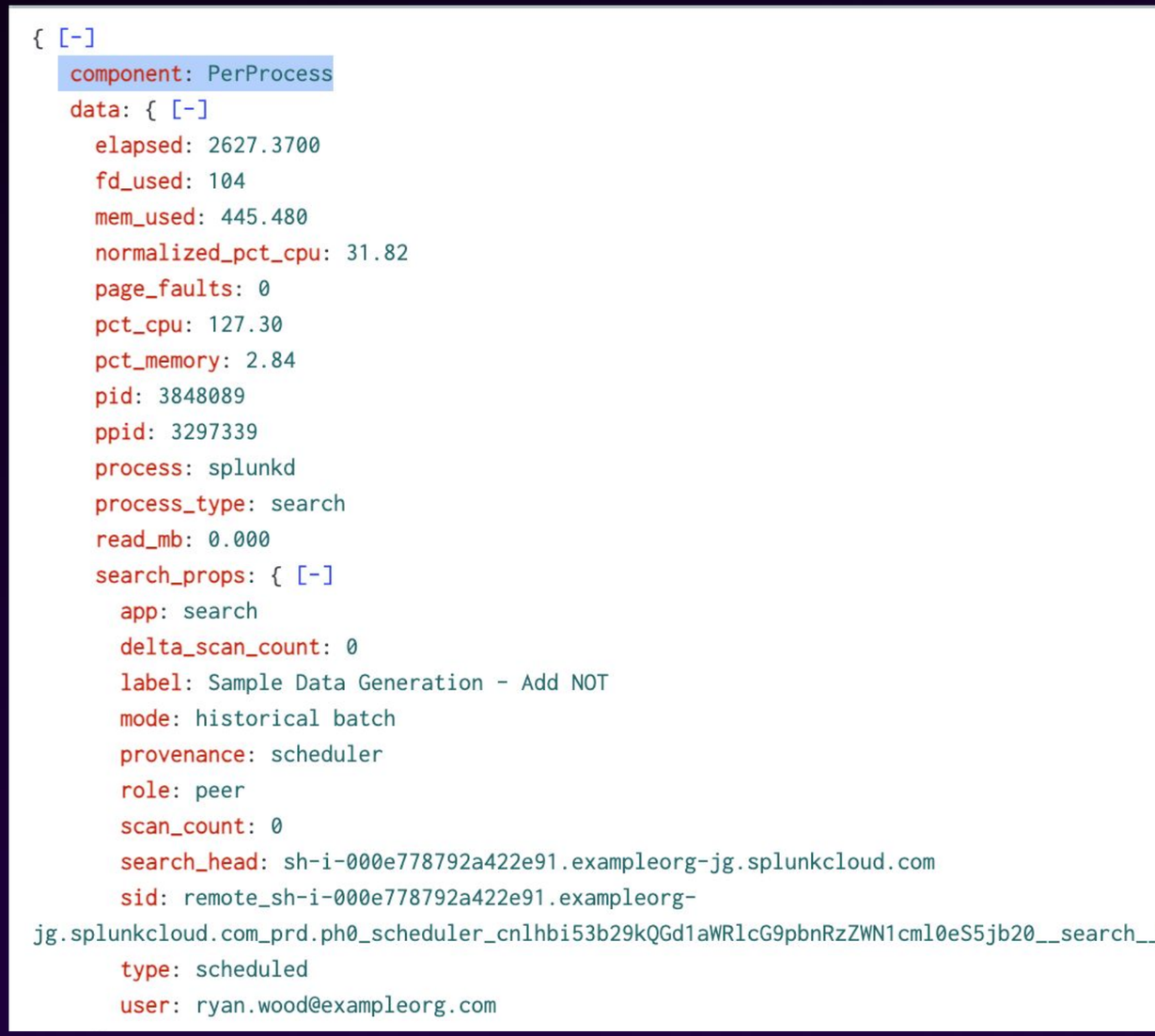


https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTintrospect#

_introspection

`sourcetype=splunk_resource_usage component=PerProcess`

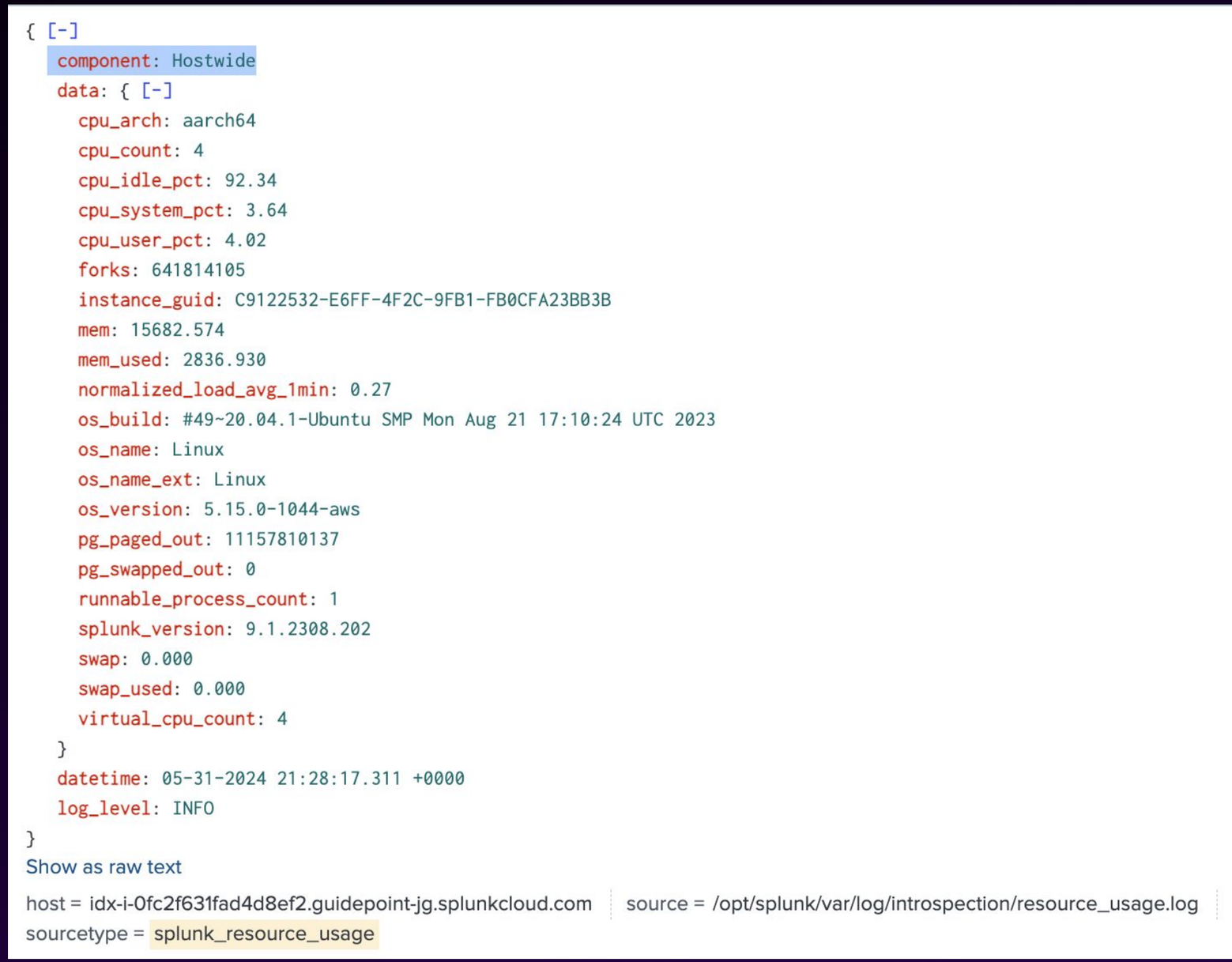
- PerProcess information
- Process must be running during collection interval
- CPU, Memory, Disk for all processes
- For Search processes, also contains search_props



_introspection

component=Hostwide

- Hostwide resource usage information
- OS info
- Powers Resource Usage Instance dashboards in Monitoring Consoles
- Collected via hostwide REST API Introspection endpoint



index=_internal

The basics

- Info related to Splunk application.
  - `sourcetype=splunkd` for troubleshooting
- Where most internal logs go
  - Except audit.log
- Default retention: 30 days
- 100+ sources
  - scheduler
  - splunkd.log
  - splunkd_access.log
  - python.log
  - metrics.log

Logging locations

The Splunk software internal logs are located in: `$SPLUNK_HOME/var/log/splunk`. This path is monitored by default, and the contents are sent to the **_internal** index. If the Splunk software is configured as a Forwarder, a subset of the logs are monitored and sent to the indexing tier.

The Splunk Introspection logs are located in `$SPLUNK_HOME/var/log/introspection`. These logs record data about the impact of the Splunk software on the host system. This path is monitored by default, and the contents are sent to the **_introspection** index. If the Splunk software is configured as a Forwarder, the monitored logs are sent to the indexing tier. See About Splunk Enterprise platform instrumentation.

The Splunk search logs are located in sub-folders under `$SPLUNK_HOME/var/run/splunk/dispatch/`. These logs record data about a search, including run time and other performance metrics. The search logs are not indexed by default. See Dispatch directory and search artifacts in the *Search Manual*.

Internal logs

A list of the internal logs in `$SPLUNK_HOME/var/log/splunk` with descriptions of their use.

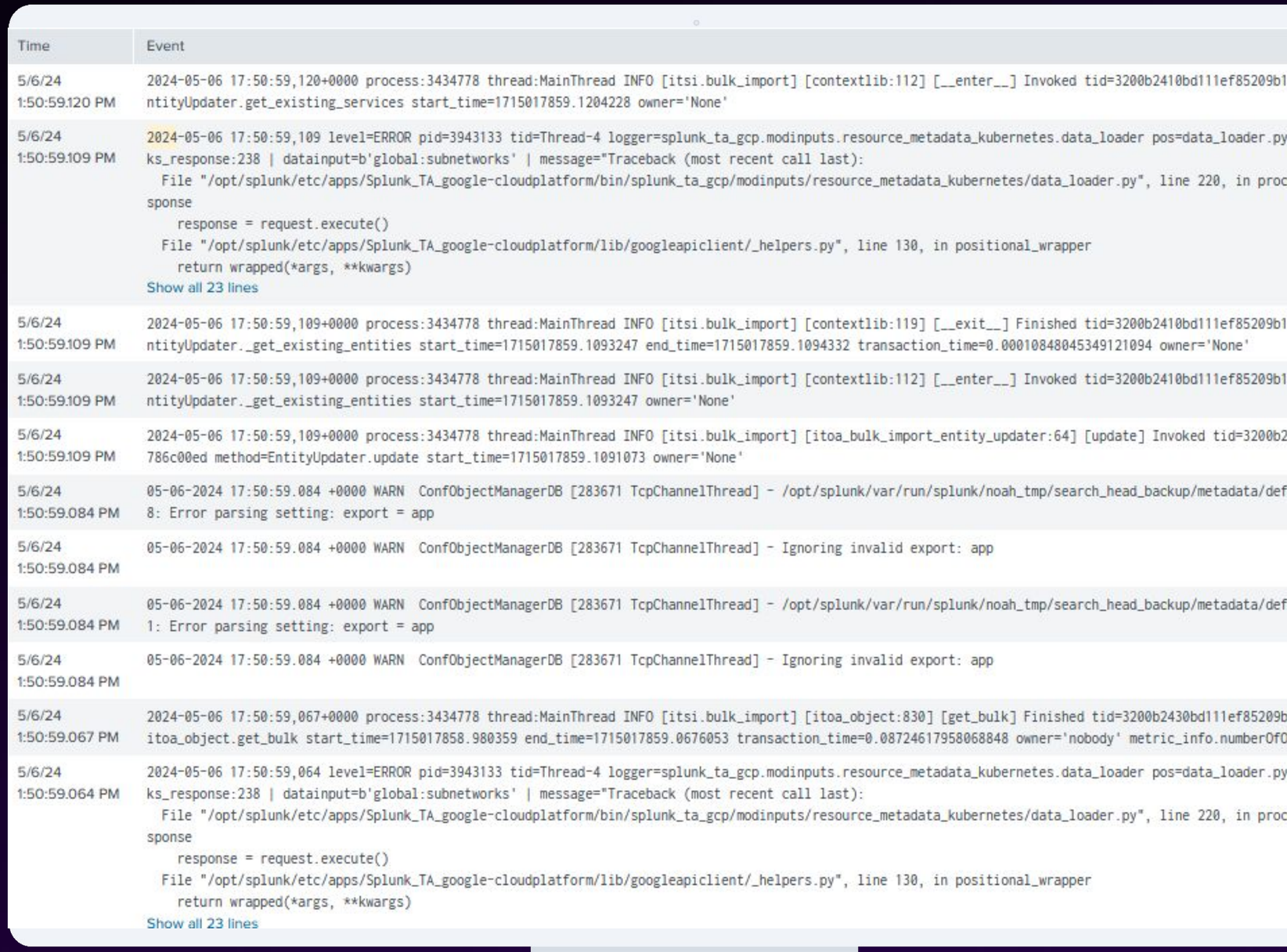
Log file name	Useful for?
audit.log	Information about user activities such as a failed or successful user log in, modifying a setting, updating a lookup file, or running a search. For example, if you're looking for information about a saved search, audit.log matches the name of a saved search (savedsearch_name) with its search ID (search_id), user, and time fields. With the search_id, you can review the logs of a specific search in the search dispatch directory. See search dispatch directory in the *Search Manual* and audit events in the *Securing Splunk Manual*. Audit.log is the only log indexed to the `_audit` index.
btool.log	A log of btool activity. See btool.
conf.log	Contains messages about configuration replication related to Search Head Clustering. See search head clustering in the *Distributed Search* manual.
configuration_change.log	Contains a record of changes to Splunk Enterprise .conf files, including the creation, updating, or deletion of new .conf files in the monitored file paths. The changes are written to the log, and indexed in the `_configtracker` index. There are options to prevent specific .conf files, and stanzas within .conf files from being monitored. To review the default tracking behavior, see Configuration Change Tracker.

https://docs.splunk.com/Documentation/Splunk/latest/Troubleshooting/WhatSplunklogsaboutitself

_internal

sourcetype=splunkd

- 80+ splunkd components
  - Metrics
  - SavedSplunker
  - HttpListener
  - PeriodicHealthReporter
  - ExecProcessor
  - TcpOutputProc (forwarders)
  - TcpInputProc
  - ModularInputs
  - UnionProcessor
  - sendmodalert



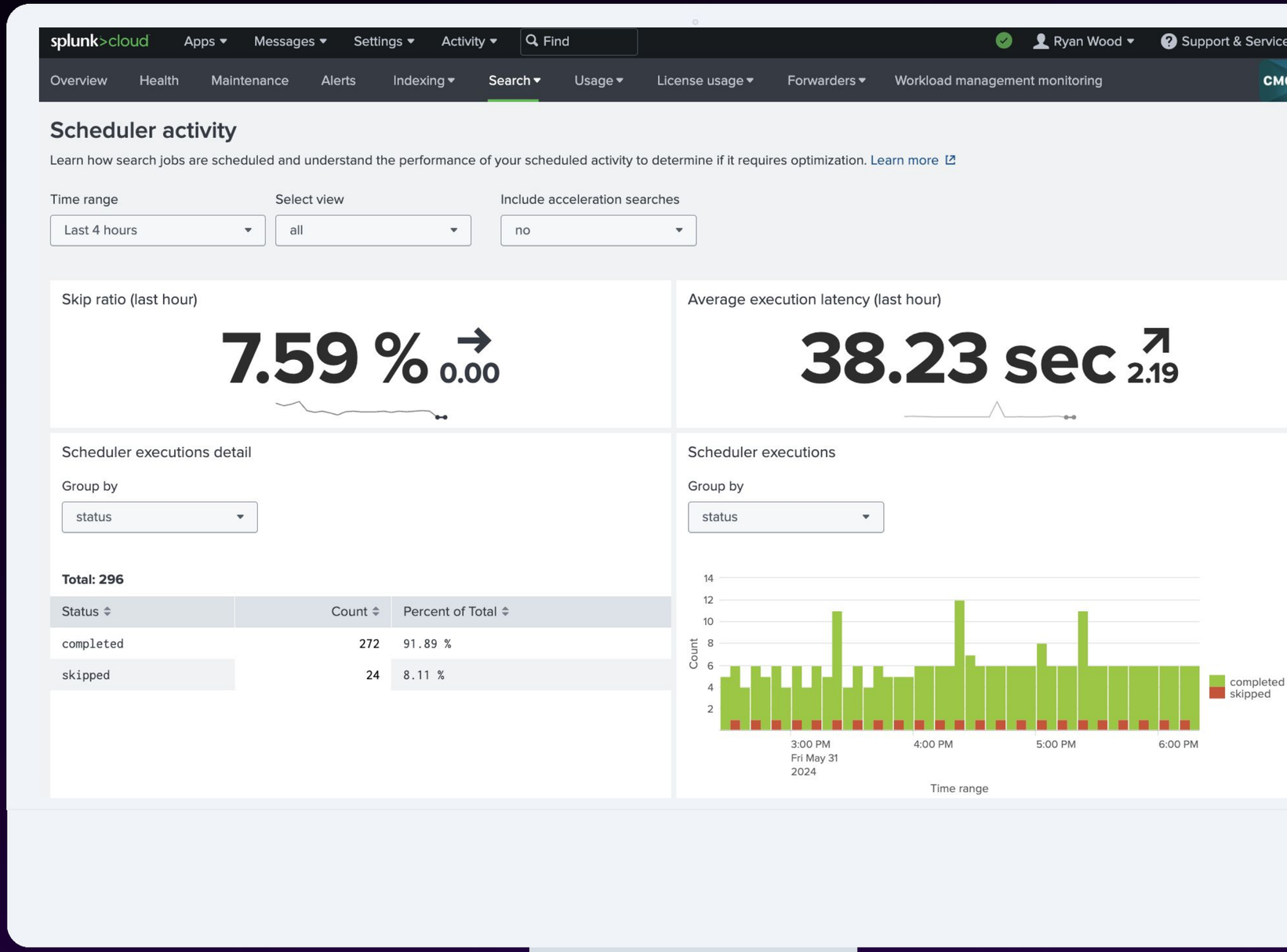
Time	Event
5/6/24 1:50:59.120 PM	2024-05-06 17:50:59,120+0000 process:3434778 thread:MainThread INFO [itsi.bulk_import] [contextlib:112] [__enter__] Invoked tid=3200b2410bd111ef85209b1 ntityUpdater.get_existing_services start_time=1715017859.1204228 owner='None'
5/6/24 1:50:59.109 PM	2024-05-06 17:50:59,109 level=ERROR pid=3943133 tid=Thread-4 logger=splunk_ta_gcp.modinputs.resource_metadata_kubernetes.data_loader pos=data_loader.py ks_response:238 | datainput=b'global:subnetworks' | message="Traceback (most recent call last): File "/opt/splunk/etc/apps/Splunk_TA_google-cloudplatform/bin/splunk_ta_gcp/modinputs/resource_metadata_kubernetes/data_loader.py", line 220, in proc sponse response = request.execute() File "/opt/splunk/etc/apps/Splunk_TA_google-cloudplatform/lib/googleapiclient/_helpers.py", line 130, in positional_wrapper return wrapped(*args, **kwargs) Show all 23 lines
5/6/24 1:50:59.109 PM	2024-05-06 17:50:59,109+0000 process:3434778 thread:MainThread INFO [itsi.bulk_import] [contextlib:119] [__exit__] Finished tid=3200b2410bd111ef85209b1 ntityUpdater._get_existing_entities start_time=1715017859.1093247 end_time=1715017859.1094332 transaction_time=0.00010848045349121094 owner='None'
5/6/24 1:50:59.109 PM	2024-05-06 17:50:59,109+0000 process:3434778 thread:MainThread INFO [itsi.bulk_import] [contextlib:112] [__enter__] Invoked tid=3200b2410bd111ef85209b1 ntityUpdater._get_existing_entities start_time=1715017859.1093247 owner='None'
5/6/24 1:50:59.109 PM	2024-05-06 17:50:59,109+0000 process:3434778 thread:MainThread INFO [itsi.bulk_import] [itoa_bulk_import_entity_updater:64] [update] Invoked tid=3200b2 786c00ed method=EntityUpdater.update start_time=1715017859.1091073 owner='None'
5/6/24 1:50:59.084 PM	05-06-2024 17:50:59.084 +0000 WARN ConfObjectManagerDB [283671 TcpChannelThread] - /opt/splunk/var/run/splunk/noah_tmp/search_head_backup/metadata/def 8: Error parsing setting: export = app
5/6/24 1:50:59.084 PM	05-06-2024 17:50:59.084 +0000 WARN ConfObjectManagerDB [283671 TcpChannelThread] - Ignoring invalid export: app
5/6/24 1:50:59.084 PM	05-06-2024 17:50:59.084 +0000 WARN ConfObjectManagerDB [283671 TcpChannelThread] - /opt/splunk/var/run/splunk/noah_tmp/search_head_backup/metadata/def 1: Error parsing setting: export = app
5/6/24 1:50:59.084 PM	05-06-2024 17:50:59.084 +0000 WARN ConfObjectManagerDB [283671 TcpChannelThread] - Ignoring invalid export: app
5/6/24 1:50:59.067 PM	2024-05-06 17:50:59,067+0000 process:3434778 thread:MainThread INFO [itsi.bulk_import] [itoa_object:830] [get_bulk] Finished tid=3200b2430bd111ef85209b itoa_object.get_bulk start_time=1715017858.980359 end_time=1715017859.0676053 transaction_time=0.08724617958068848 owner='nobody' metric_info.numberOfO
5/6/24 1:50:59.064 PM	2024-05-06 17:50:59,064 level=ERROR pid=3943133 tid=Thread-4 logger=splunk_ta_gcp.modinputs.resource_metadata_kubernetes.data_loader pos=data_loader.py ks_response:238 | datainput=b'global:subnetworks' | message="Traceback (most recent call last): File "/opt/splunk/etc/apps/Splunk_TA_google-cloudplatform/bin/splunk_ta_gcp/modinputs/resource_metadata_kubernetes/data_loader.py", line 220, in proc sponse response = request.execute() File "/opt/splunk/etc/apps/Splunk_TA_google-cloudplatform/lib/googleapiclient/_helpers.py", line 130, in positional_wrapper return wrapped(*args, **kwargs) Show all 23 lines

https://docs.splunk.com/Documentation/Splunk/latest/Troubleshooting/WhatSplunklogsaboutitself

_internal

sourcetype=scheduler

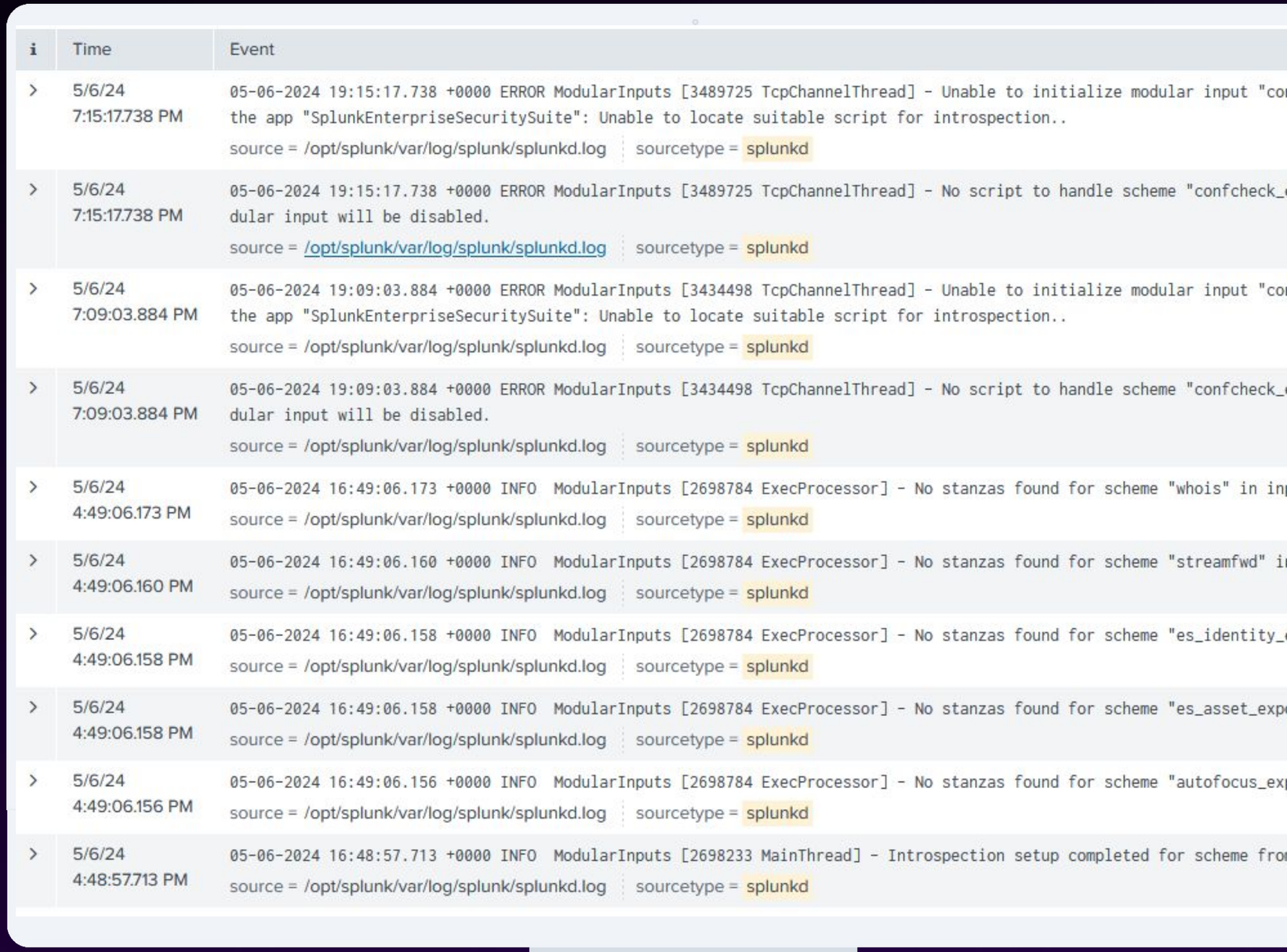
- scheduler.log
  - All actions (successful or unsuccessful) performed by the splunkd search and alert scheduler.
  - Typically shows scheduled search activity or errors.
- Scheduled search activity
  - Includes alerts and datamodel accelerations
- Used to populate "Scheduler Activity", "Skipped Searches" panels in CMC



_internal

The basics

- modularInputs
  - Messages from modular inputs
  - Errors usually mean the MI isn't running and needs attention
  - index = _internal source = *splunkd.log sourcetype = splunkd component = ModularInputs

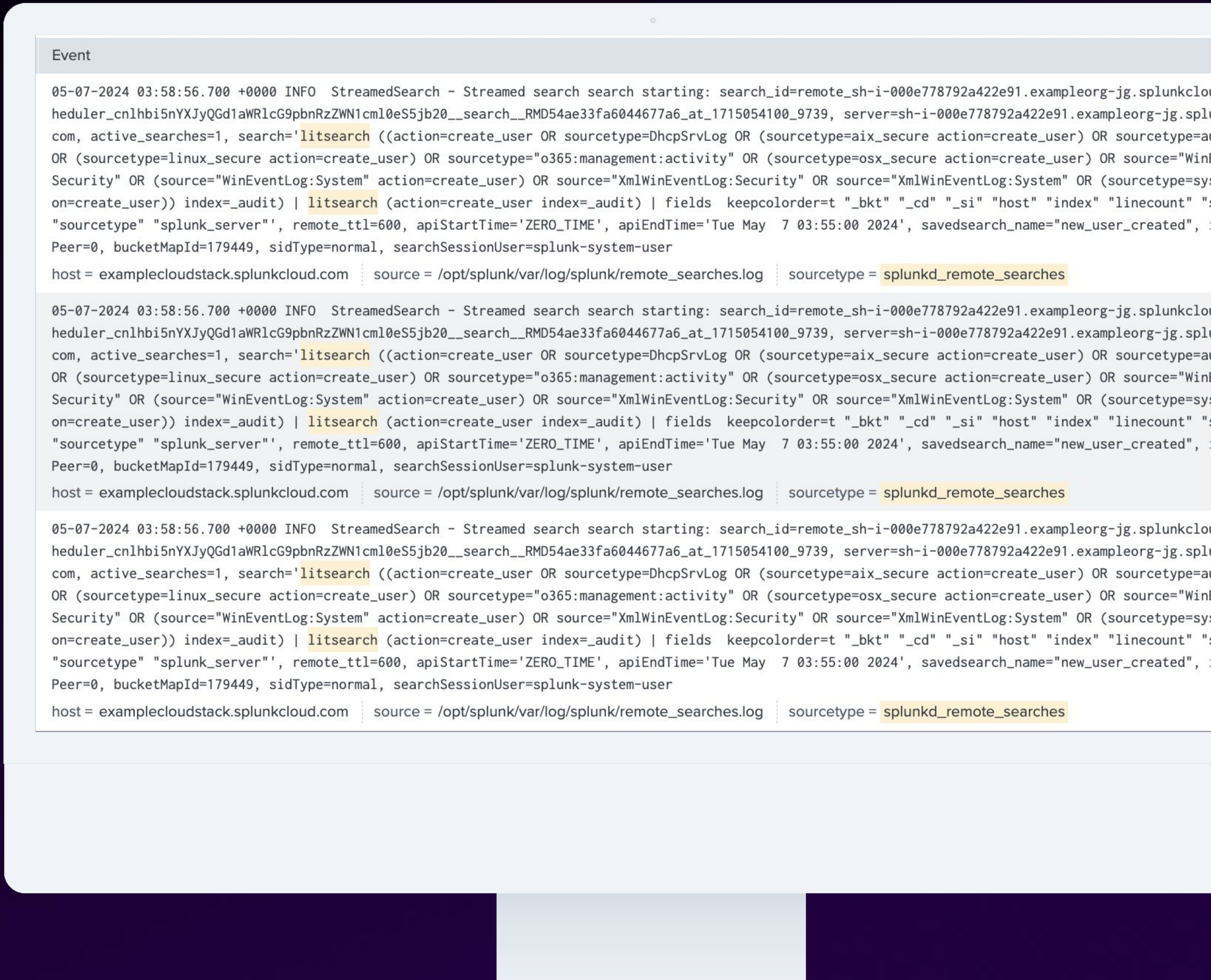


i	Time	Event
>	5/6/24 7:15:17.738 PM	05-06-2024 19:15:17.738 +0000 ERROR ModularInputs [3489725 TcpChannelThread] - Unable to initialize modular input "co the app "SplunkEnterpriseSecuritySuite": Unable to locate suitable script for introspection.. source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 7:15:17.738 PM	05-06-2024 19:15:17.738 +0000 ERROR ModularInputs [3489725 TcpChannelThread] - No script to handle scheme "confcheck_ dular input will be disabled. source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 7:09:03.884 PM	05-06-2024 19:09:03.884 +0000 ERROR ModularInputs [3434498 TcpChannelThread] - Unable to initialize modular input "co the app "SplunkEnterpriseSecuritySuite": Unable to locate suitable script for introspection.. source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 7:09:03.884 PM	05-06-2024 19:09:03.884 +0000 ERROR ModularInputs [3434498 TcpChannelThread] - No script to handle scheme "confcheck_ dular input will be disabled. source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 4:49:06.173 PM	05-06-2024 16:49:06.173 +0000 INFO ModularInputs [2698784 ExecProcessor] - No stanzas found for scheme "whois" in inp source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 4:49:06.160 PM	05-06-2024 16:49:06.160 +0000 INFO ModularInputs [2698784 ExecProcessor] - No stanzas found for scheme "streamfwd" i source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 4:49:06.158 PM	05-06-2024 16:49:06.158 +0000 INFO ModularInputs [2698784 ExecProcessor] - No stanzas found for scheme "es_identity_ source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 4:49:06.158 PM	05-06-2024 16:49:06.158 +0000 INFO ModularInputs [2698784 ExecProcessor] - No stanzas found for scheme "es_asset_exp source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 4:49:06.156 PM	05-06-2024 16:49:06.156 +0000 INFO ModularInputs [2698784 ExecProcessor] - No stanzas found for scheme "autofocus_ex source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd
>	5/6/24 4:48:57.713 PM	05-06-2024 16:48:57.713 +0000 INFO ModularInputs [2698233 MainThread] - Introspection setup completed for scheme fro source = /opt/splunk/var/log/splunk/splunkd.log sourcetype = splunkd

_internal

remote_searches.log

- Streamed search activity between SH and indexers
- Captures information from Peer for searches
- Contains literal search query run, with all macros and eventtypes expanded



remote_searches.log Additional Detail

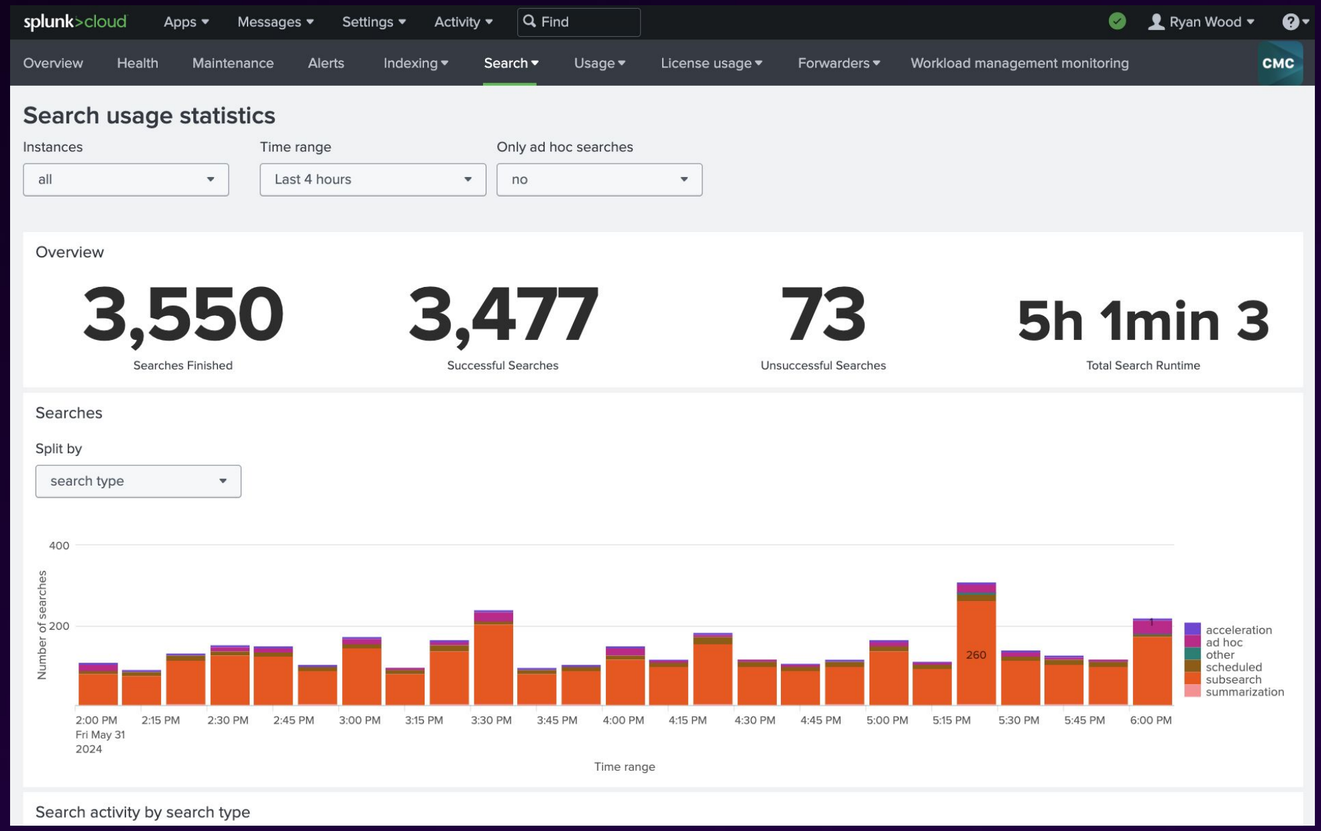
Additional information related to `_internal sourcetype=splunkd_remote_searches`

- Remote searches run on indexer peers providing data for the search
- Separate SIDs are generated for subsearches, search phases, etc - normalization is needed to unify
- If the search does not specify source or sourcetype:
  - *litSearch* entries in remote_searches.log events will include a bunch of potential eventtype match logic. This inflates the event and adds a bunch of SPL that wasn't explicitly in the original query run.
- When queries are sent to the search peer, the peer performs the initial "map" part of the search to return to the SH, which is captured and logged with these remote_population* prefix SIDs.
  - Base events reporting on search process do not include remote_population metrics in them
  - Remote_population* SID event data should be normalized and aggregated alongside the standard SID events when doing reporting on the search process

index=_audit

The basics

- Information about user activity, including running searches
  - Contains search info AND SPL queries
- Default retention: 6 years (3 years in Splunk Cloud)
- Used to populate the Search Usage Statistics and parts of the Expensive Searches dashboards in CMC

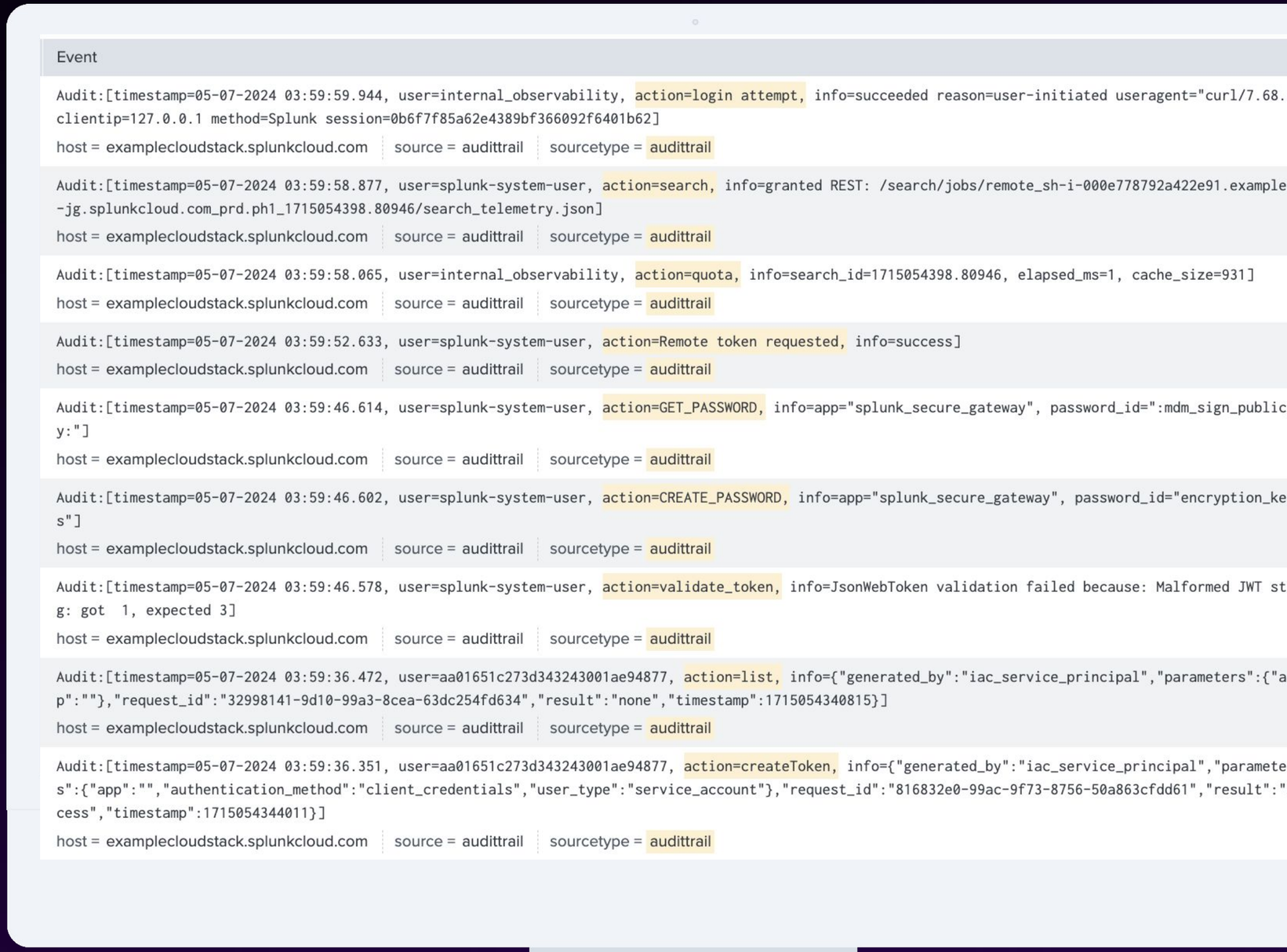


https://docs.splunk.com/Documentation/Splunk/latest/Troubleshooting/WhatSplunklogsaboutitself#

audittrail

The basics

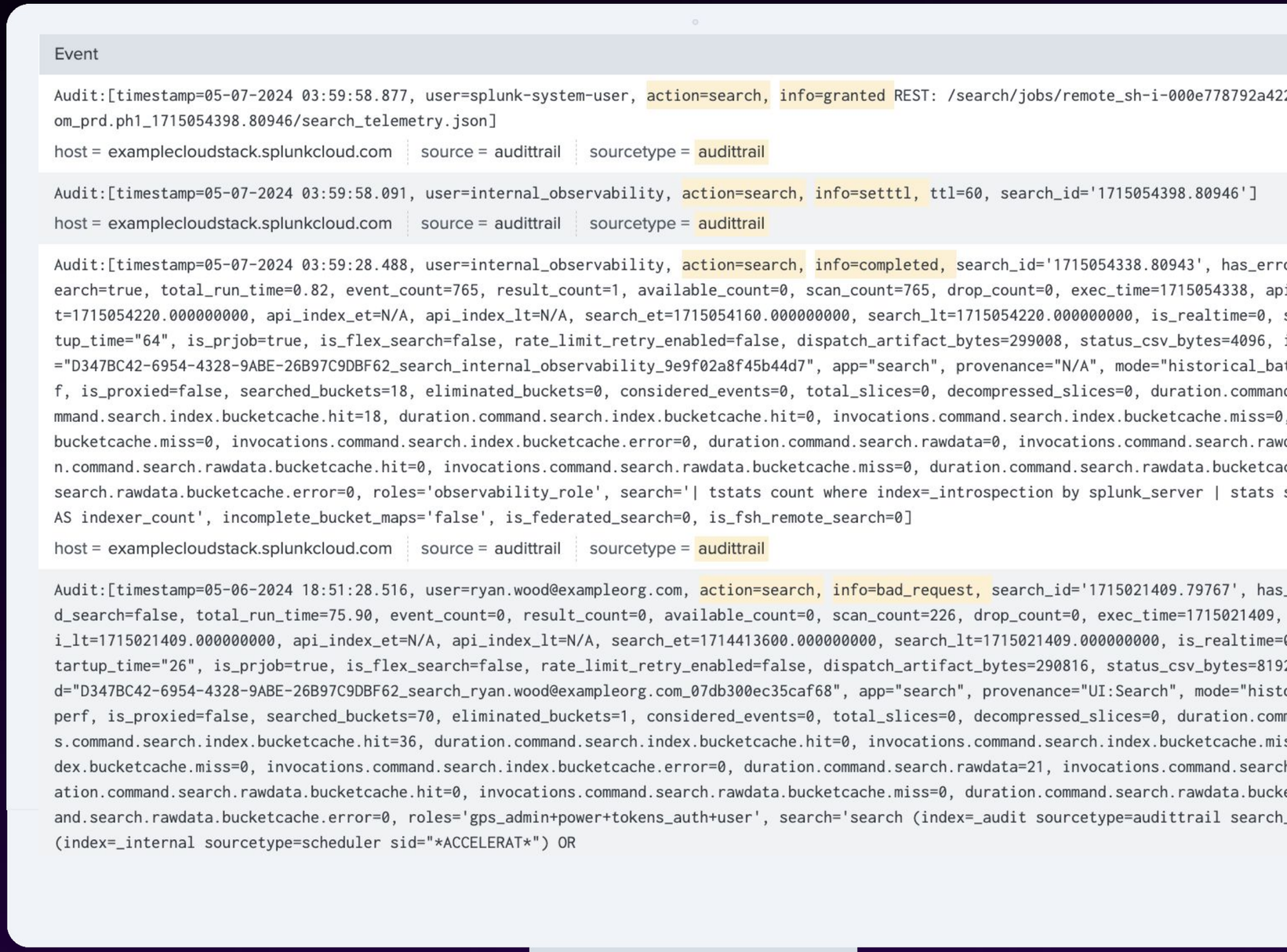
- Dozens of “actions” are logged
- Interesting actions
  - search
  - create_user
  - create_saved_search
  - alert_fired
  - login attempt
  - remote_bucket_download



audittrail

action=search

- audittrail contains the most verbose event data on searches
- SPL run in addition to info on status
- Includes REST API calls when searches are initiated
- *info* field can be used for troubleshooting
  - ie `info=bad_request`

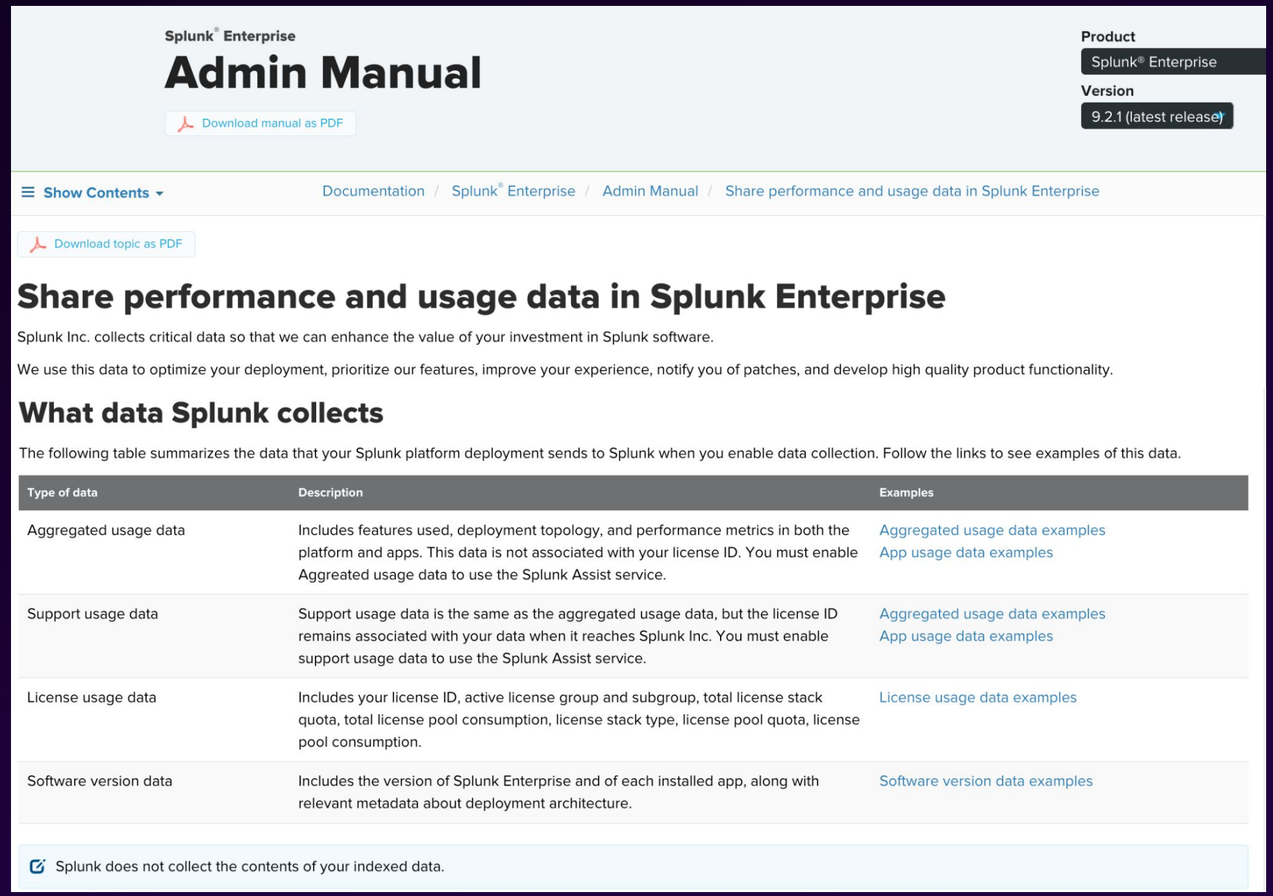


index=_telemetry

The basics

- Instrumentation info collected from internal data sources
  - Features, usage
  - Search types, command usage
  - License usage
- Default retention: 2 years
  - Data size limit: 256mb
- Event Format depends on Platform
  - Enterprise vs Cloud

https://docs.splunk.com/Documentation/Splunk/latest/Admin/Shareperformancedata#



Splunk® Enterprise

Admin Manual

Download manual as PDF

Product: Splunk® Enterprise

Version: 9.2.1 (latest release)

Show Contents

Documentation / Splunk® Enterprise / Admin Manual / Share performance and usage data in Splunk Enterprise

Download topic as PDF

Share performance and usage data in Splunk Enterprise

Splunk Inc. collects critical data so that we can enhance the value of your investment in Splunk software.

We use this data to optimize your deployment, prioritize our features, improve your experience, notify you of patches, and develop high quality product functionality.

What data Splunk collects

The following table summarizes the data that your Splunk platform deployment sends to Splunk when you enable data collection. Follow the links to see examples of this data.

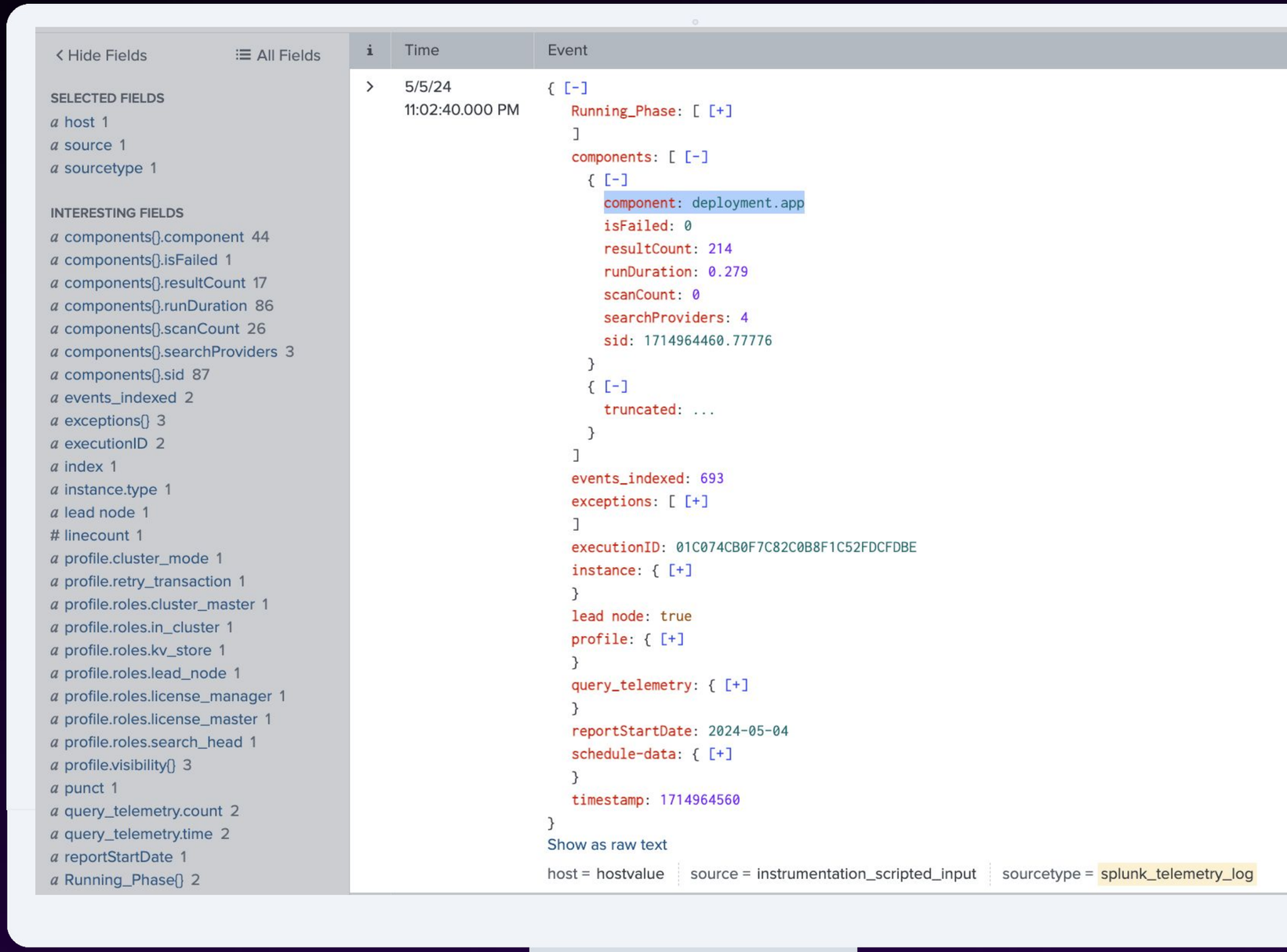
Type of data	Description	Examples
Aggregated usage data	Includes features used, deployment topology, and performance metrics in both the platform and apps. This data is not associated with your license ID. You must enable Aggreated usage data to use the Splunk Assist service.	Aggregated usage data examples App usage data examples
Support usage data	Support usage data is the same as the aggregated usage data, but the license ID remains associated with your data when it reaches Splunk Inc. You must enable support usage data to use the Splunk Assist service.	Aggregated usage data examples App usage data examples
License usage data	Includes your license ID, active license group and subgroup, total license stack quota, total license pool consumption, license stack type, license pool quota, license pool consumption.	License usage data examples
Software version data	Includes the version of Splunk Enterprise and of each installed app, along with relevant metadata about deployment architecture.	Software version data examples

Splunk does not collect the contents of your indexed data.

_telemetry

The basics

- *sourcetype IN (splunk_telemetry, splunk_telemetry_log)*
- Info related to telemetry jobs execution
- Collection jobs identified by "component" value

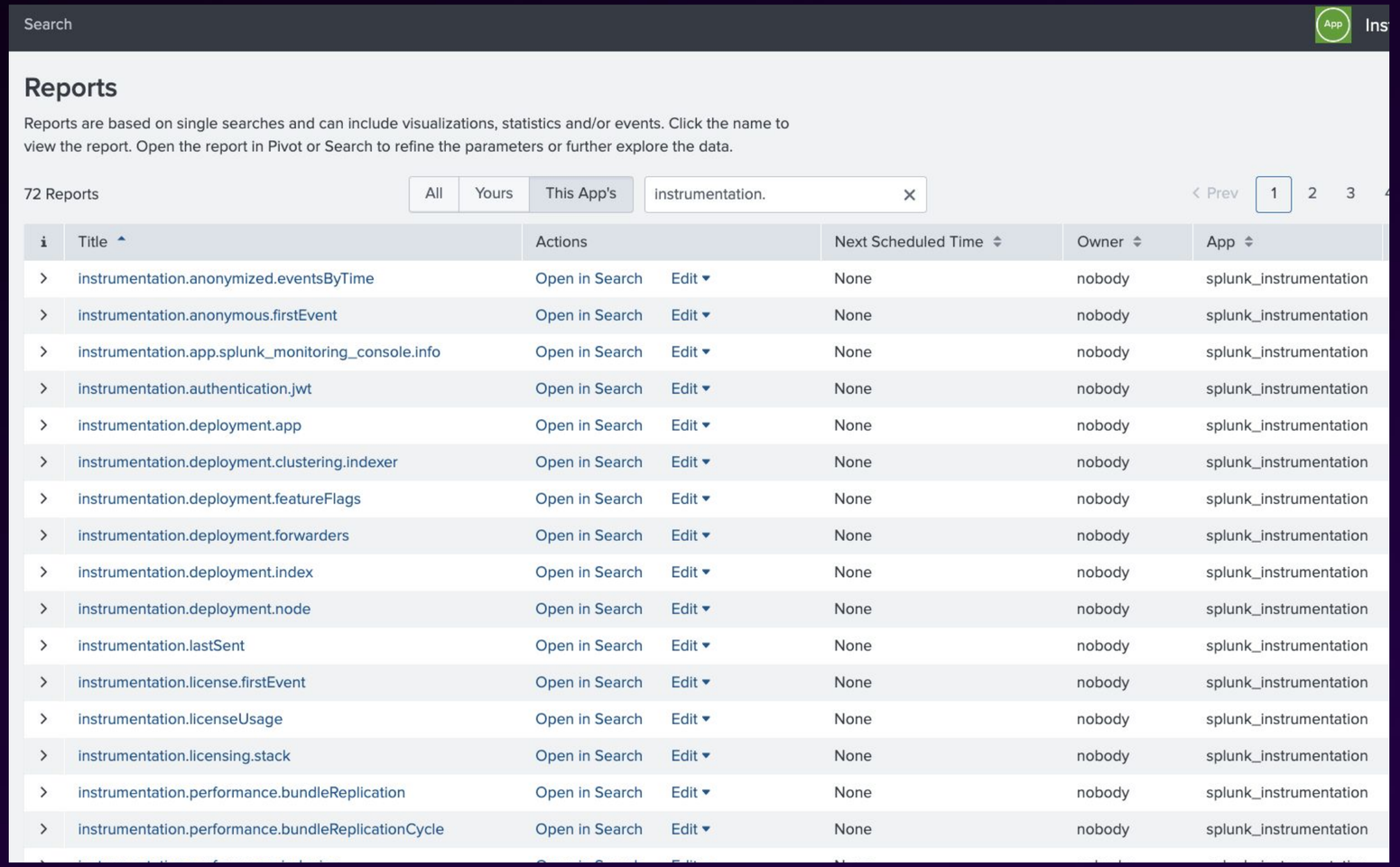


https://docs.splunk.com/Documentation/Splunk/latest/Admin/Shareperformancedata#

_telemetry (instrumentation)

app=splunk_instrumentation

- Core app default in all Splunk
- Contains savedsearches used to collect telemetry data
  - Jobs run daily at 3am by default
- Runs in both Enterprise & Cloud, but does not write data to `index=_telemetry` in Enterprise
  - Data instead written to `index=_introspection`



* Docs contain examples of all components.

https://docs.splunk.com/Documentation/Splunk/latest/Admin/Shareperformancedata#Examples_of_data_sent_to_Splunk

_telemetry (instrumentation)

app=splunk_instrumentation

- Core app default in all Splunk
- Contains savedsearches used to collect telemetry data
  - Jobs run daily at 3am by default
- Runs in both Enterprise & Cloud, but does not write data to `index=_telemetry` in Enterprise
  - Data instead written to `index=_introspection`

Aggregated usage data examples

The following example demonstrates the data sent to Splunk when sharing of aggregated usage data is enabled.

Component	Description	Example
`usage.search.searchtelemetry.sourcetypeUsage`	Sourcetype usage.	{ [-] sourcetypeUsage: [[-] { [-] http_event_collector_metrics: 1 kvstore: 1 mongod: 3 search_telemetry: 1 splunk_disk_objects: 1 splunk_resource_usage: 1 splunk_web_service: 3 splunkd: 11 splunkd_remote_searches: 3 splunkd_ui_access: 2 }] }

* Docs contain examples of all components.

https://docs.splunk.com/Documentation/Splunk/latest/Admin/Shareperformancedata#Examples_of_data_sent_to_Splunk

Instrumentation Data

Located in different indexes based on your environment type.

Splunk® Cloud Platform

`index=_telemetry sourcetype=splunk_cloud_telemetry`

- Same data as Enterprise, but nested within JSON.
- Much longer retention
- Additional JSON wrapper means rename is needed:

```
| rename data.* AS *
```

Splunk® Enterprise

`index=_introspection sourcetype=splunk_telemetry`

- Much shorter retention
- Must be written to secondary index for longer retention
  - SPL to do this in PDF slides

Instrumentation Data

Located in different indexes based on your environment type.

Splunk® Cloud Platform

`index=_telemetry sourcetype=splunk_cloud_telemetry`

- Same data as Enterprise, but nested within JSON.


```
{ [-]
  component: TelemetryCloudData
  data: { [-]
    component: usage.search.searchtelemetry.sourcetypeUsage
    data: { [-]
      sourcetypeUsage: [ [-]
        { [-]
          audittrail: 350
          search_log_events: 889
          splunk_resource_usage: 206
          truncated: ...
        }
      ]
    }
    date: 2024-05-05
    deploymentID: CLOUD-3e530e68d52f4b3250ebf1ac98bf44b29a48d22f07d70e5b8bb7cee59a1393bf
    executionID: 01C074CB0F7C82C0B8F1C52FDCFDBE
    timestamp: 1714964466
    transactionID: 22D32667-589A-AA93-DB57-0605D9FE94F9
    version: 4
    visibility: anonymous,support
  }
  datetime: 2024-05-06 03:02:40,496
  log_level: INFO
}
```


Show as raw text

host = hostvalue | source = splunk_instrumentation_cloud.log | sourcetype = splunk_cloud_telemetry

Splunk® Enterprise

`index=_introspection sourcetype=splunk_telemetry`


```
{ [-]
  component: usage.search.searchtelemetry.sourcetypeUsage
  data: { [-]
    sourcetypeUsage: [ [-]
      { [-]
        audittrail: 210
        scheduler: 176
        stash: 729
        truncated: ...
      }
    ]
  }
  date: 2024-05-04
  executionID: 0C671A090B3E117CB35D3E885518C3
  timestamp: 1714950102
  visibility: anonymous,support
}
```


Show as raw text

host = hostvalue | source = http-stream | sourcetype = splunk_telemetry

splunk_instrumentation Additional Detail

Additional information related to instrumentation app

- Instrumentation jobs are launched via scheduled python job, which is why savedsearch objects are not scheduled in UI.

- All Instrumentation jobs contain terms:
  - executionID
  - component

- Details for each instrumentation component can be found in docs: https://docs.splunk.com/Documentation/Splunk/9.2.1/Admin/Shareperformancedata#Examples_of_data_sent_to_Splunk

- Scheduling and configurations for telemetry collection are defined in *telemetry.conf* https://docs.splunk.com/Documentation/Splunk/9.2.1/Admin/Telemetryconf
  - Whether to send each type of data

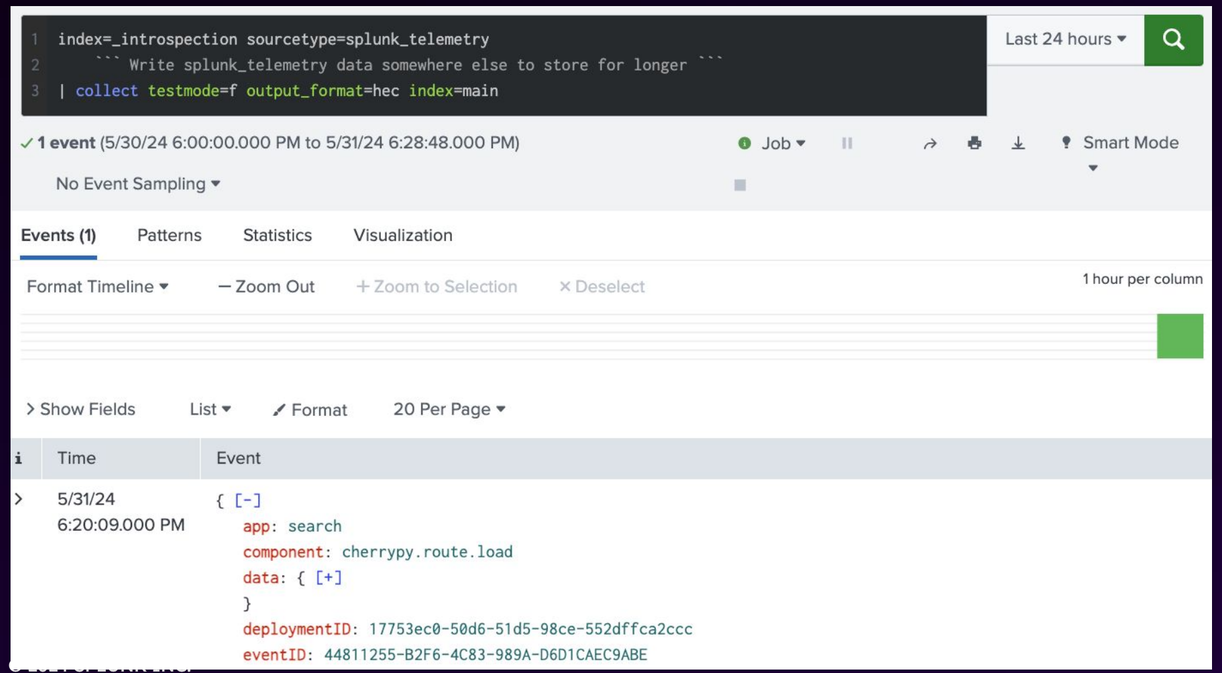
Instrumentation - Saving Job Data for Longer Retention in Splunk Enterprise

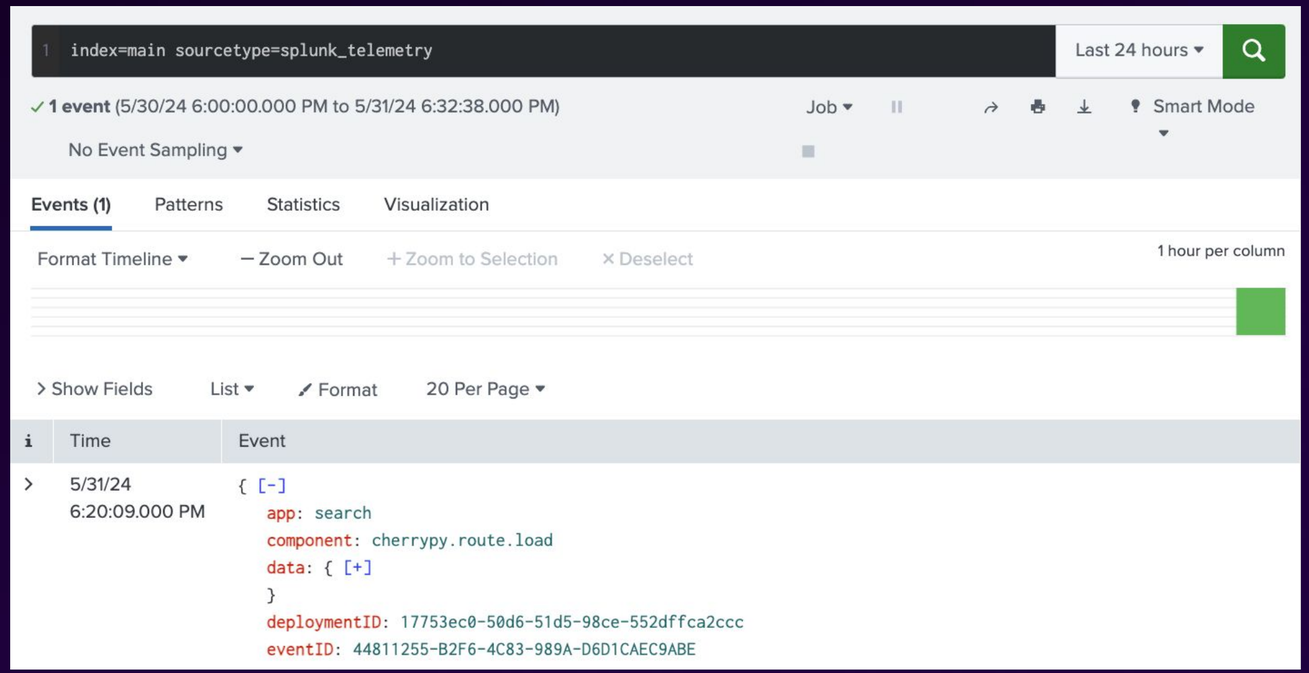
Splunk Enterprise Only

Straightforward SPL for populating _telemetry with what's stored in introspection

- Use collect with output_format=hec to retain the original host, source, sourcetype values

```
index=_introspection sourcetype=splunk_telemetry
    ``` Write splunk_telemetry data somewhere else to store for longer ```
| collect testmode=f output_format=hec index=<target index>
```




```

Bonus: _configtracker

Records configuration changes across the splunk environment

Track changes to configs

- Reports changes to .conf files made via the UI
- Does not attribute changes to a specific user, just notes that it was changed
- This example event shows the the savedsearch "Sample Data Generation - Add NOT" having it's schedule enabled.

Event


```
{ [-]
   component: ConfigChange
   data: { [-]
     action: update
     changes: [ [-]
       { [-]
         properties: [ [-]
           { [-]
             name: enableSched
             new_value:
             old_value: 1
           }
         ]
         stanza: Sample Data Generation - Add NOT
       }
     ]
     epoch_time: 1717122356
     modtime: Thu May 30 22:25:51 2024
     new_checksum: 0x2a0deaa325d1ce21
     old_checksum: 0x66f157ff28b5f08c
     path: /opt/splunk/etc/users/admin/search/local/savedsearches.conf
   }
   datetime: 05-30-2024 22:25:56.925 -0400
   log_level: INFO
}
```


Show as raw text

host = examplehost | source = /opt/splunk/var/log/splunk/configuration_change.log | sourcetype = splunk_configuration_change

But how can we *use* it?

Usecases & Examples

.conf24

splunk>

Bring on the Usecases

Tips, Common Questions, Etc

Heads ups, Gotchas, Tips, Warnings, Requirements, Etc. related to the internal data sources available.

- Audittrail events automatically extract key=value fields, including those embedded in the search field.
- Don’t assume anything is 100%
- Be aware of index retention times.
- Long search time ranges with SmartStore indexes may be slower because of cache misses.

Usecase - Identifying Problematic Searches

How can I find broken, inefficient, or problematic searches?

Start with utilities already available!

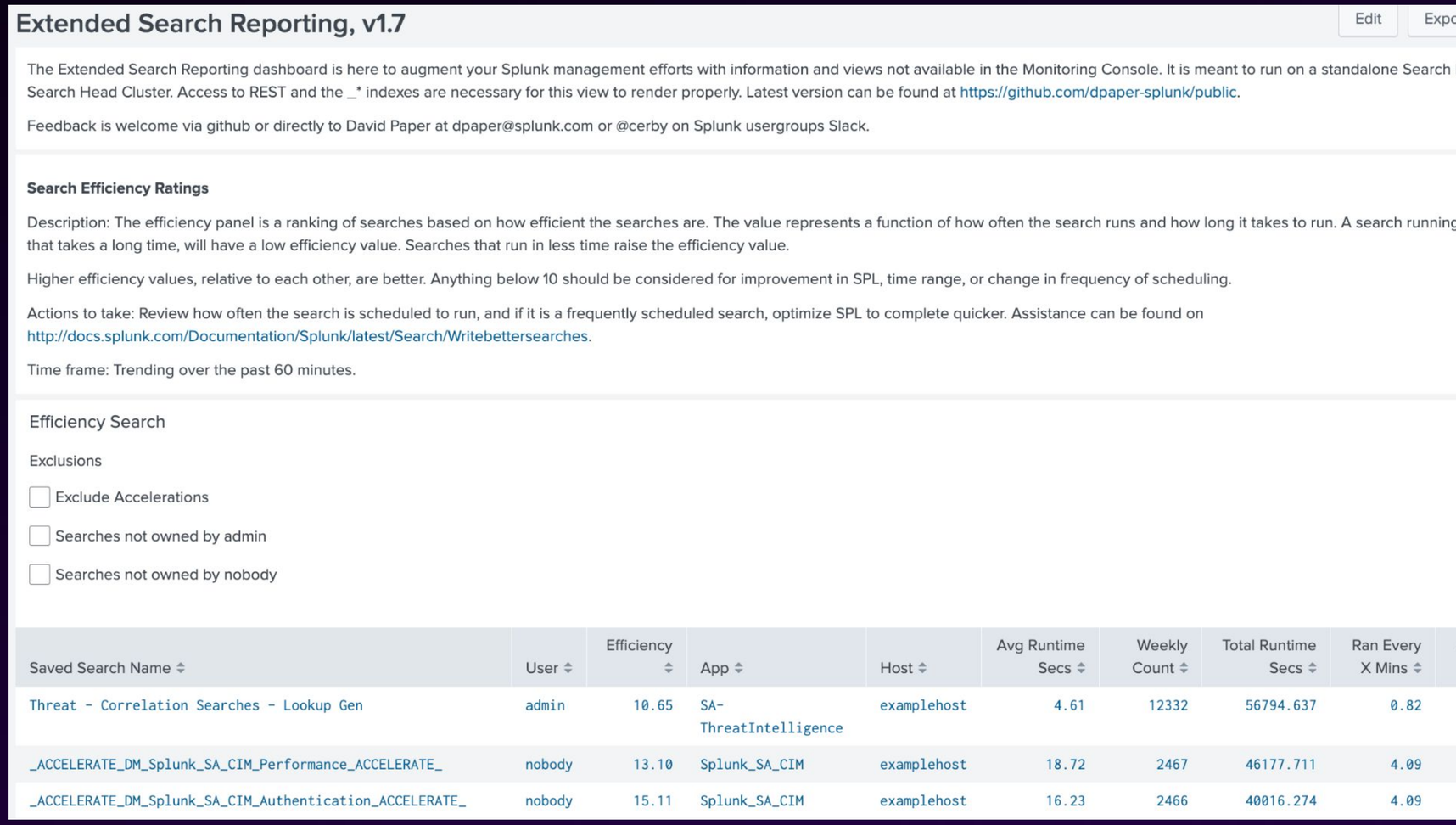
© 2024 SPLUNK INC.

Usecase - Identifying Problematic Searches

Extended Search Reporting Dashboard

Extended Search Reporting
Credit: David Paper

- Search Efficiency
- Search Improvements
- Events Scanned vs Returned



Extended Search Reporting, v1.7

Edit Expo

The Extended Search Reporting dashboard is here to augment your Splunk management efforts with information and views not available in the Monitoring Console. It is meant to run on a standalone Search Search Head Cluster. Access to REST and the _* indexes are necessary for this view to render properly. Latest version can be found at https://github.com/dpaper-splunk/public.

Feedback is welcome via github or directly to David Paper at dpaper@splunk.com or @cerby on Splunk usergroups Slack.

Search Efficiency Ratings

Description: The efficiency panel is a ranking of searches based on how efficient the searches are. The value represents a function of how often the search runs and how long it takes to run. A search running that takes a long time, will have a low efficiency value. Searches that run in less time raise the efficiency value.

Higher efficiency values, relative to each other, are better. Anything below 10 should be considered for improvement in SPL, time range, or change in frequency of scheduling.

Actions to take: Review how often the search is scheduled to run, and if it is a frequently scheduled search, optimize SPL to complete quicker. Assistance can be found on http://docs.splunk.com/Documentation/Splunk/latest/Search/Writebettersearches.

Time frame: Trending over the past 60 minutes.

Efficiency Search

Exclusions

- [ ] Exclude Accelerations
- [ ] Searches not owned by admin
- [ ] Searches not owned by nobody

Saved Search Name	User	Efficiency	App	Host	Avg Runtime Secs	Weekly Count	Total Runtime Secs	Ran Every X Mins
Threat - Correlation Searches - Lookup Gen	admin	10.65	SA-ThreatIntelligence	examplehost	4.61	12332	56794.637	0.82
_ACCELERATE_DM_Splunk_SA_CIM_Performance_ACCELERATE_	nobody	13.10	Splunk_SA_CIM	examplehost	18.72	2467	46177.711	4.09
_ACCELERATE_DM_Splunk_SA_CIM_Authentication_ACCELERATE_	nobody	15.11	Splunk_SA_CIM	examplehost	16.23	2466	40016.274	4.09

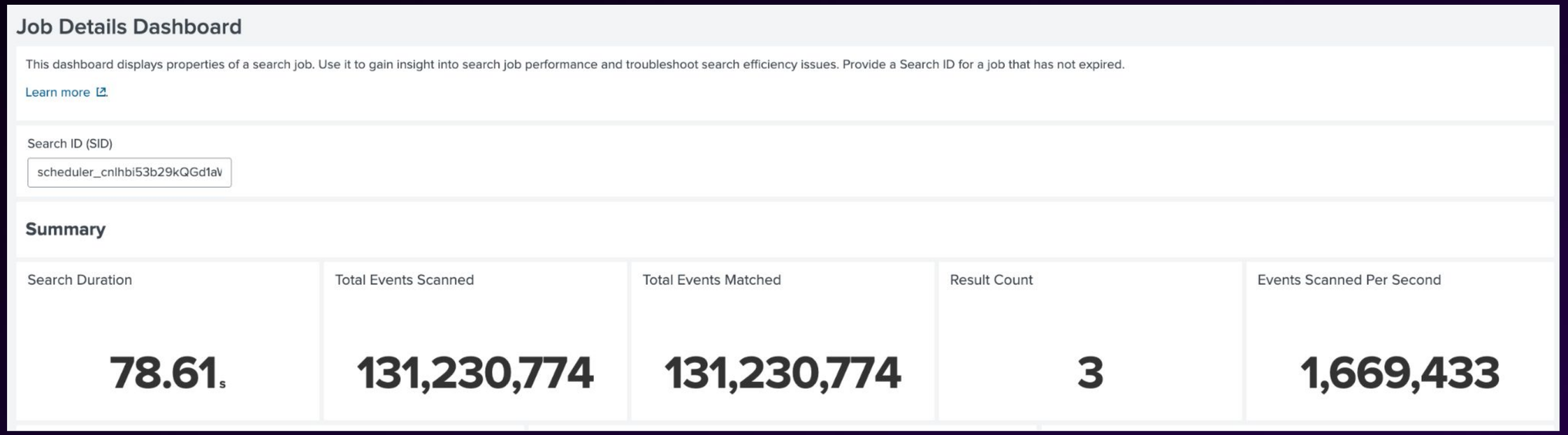
*Credit to David Paper
https://github.com/dpaper-splunk/public/blob/master/dashboards/extended_search_reporting.xml

© 2024 SPLUNK INC.

Usecase - Identifying Problematic Searches

Job Details Dashboard

Access via:
Job Inspector > Job Details



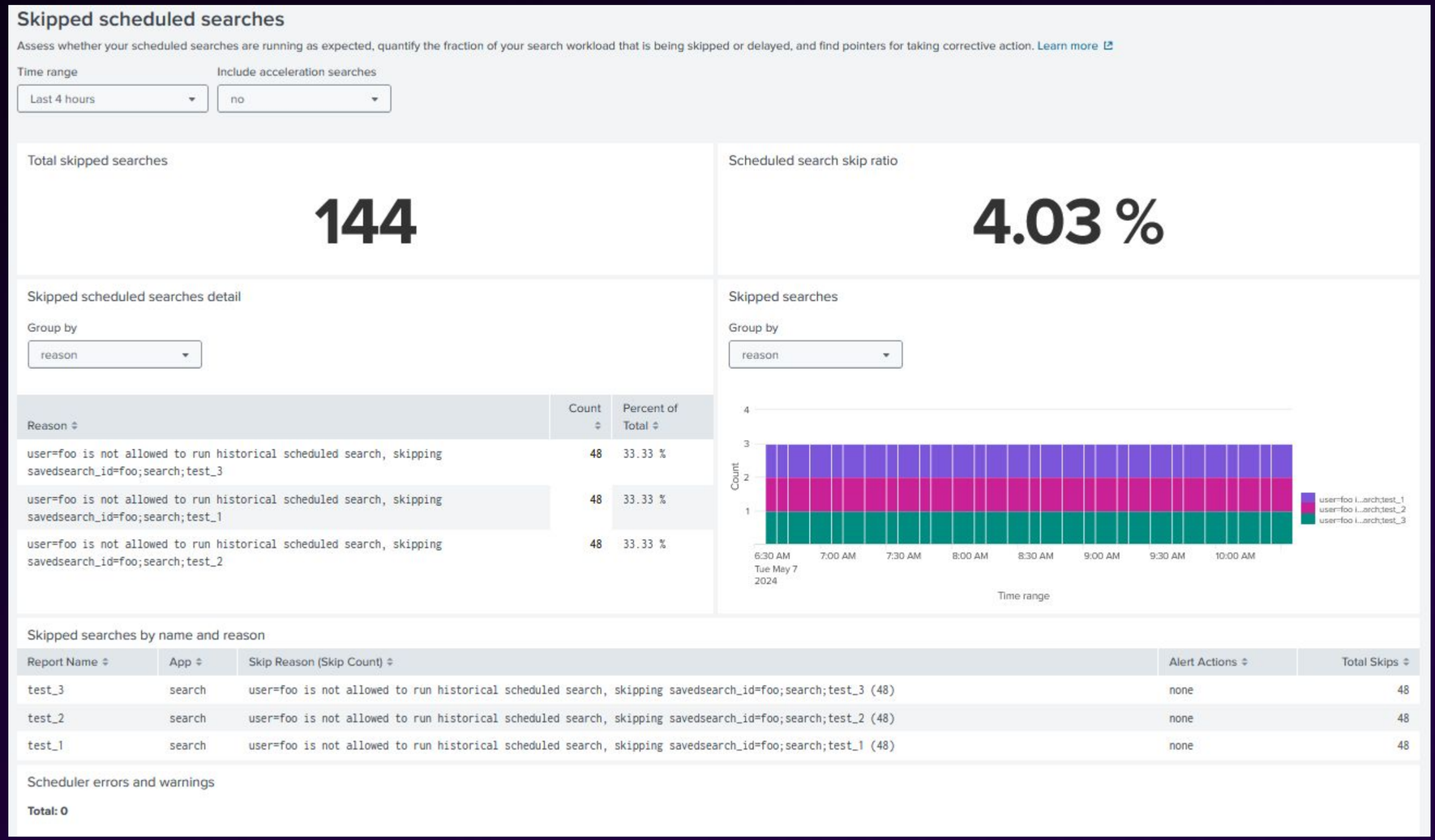
Splunk Docs on the Job Details dashboard:

https://docs.splunk.com/Documentation/Splunk/latest/Search/ViewsearchjobpropertieswiththeJobInspector#Open_the_Job_Details_dashboard_to_get_a_concise_overview_of_your_search_job

Usecase - Identifying Problematic Searches

Cloud Monitoring Console > Search > Skipped scheduled searches

For Splunk Enterprise:
Scheduler Activity: Instance



Usecase - Identifying Problematic Searches

```
_internal sourcetype=scheduler
```

Report on skipped searches using…

savedsearch_name - Search Name

status - Search status

reason - Reason for status

app - Which app search ran in

user - Who ran the search

Event

05-09-2024 02:43:56.057 +0000 INFO SavedSplunker - savedsearch_id="ryan.garr@exampleorg.com;search;new_user_created", search_type="scheduled", user="ryan.garr@exampleorg.com", app="search", savedsearch_name="new_user_created", priority=default, status=skipped, reason="The maximum number of concurrent running jobs for this historical scheduled search on this instance has been reached", concurrency_category="historical_scheduled", concurrency_context="saved-search_instance-wide", concurrency_limit=1, scheduled_time=1715222636, window_time=-1, skipped_count=1, filtered_count=0

host = examplecloudstack.splunkcloud.com | source = /opt/splunk/var/log/splunk/scheduler.log | sourcetype = scheduler

05-09-2024 02:42:31.005 +0000 INFO SavedSplunker - savedsearch_id="nobody;search;Sample Data Generation - Add NOT", search_type="scheduled", user="ryan.wood@exampleorg.com", app="search", savedsearch_name="Sample Data Generation - Add NOT", priority=default, status=skipped, reason="The maximum number of concurrent running jobs for this historical scheduled search on this instance has been reached", concurrency_category="historical_scheduled", concurrency_context="saved-search_instance-wide", concurrency_limit=1, scheduled_time=1715222550, window_time=-1, skipped_count=1, filtered_count=0

host = examplecloudstack.splunkcloud.com | source = /opt/splunk/var/log/splunk/scheduler.log | sourcetype = scheduler

Skipped Searches by…

Using _internal sourcetype=scheduler

Simple Timechart view of skipped searches by savedsearch_name

```
index=_internal sourcetype=scheduler status=skipped
| timechart count by savedsearch_name
```

Simple view of skipped searches by reason

```
index=_internal sourcetype=scheduler status=skipped reason=*
| stats count, values(reason) AS reason by user, app, savedsearch_name
```

© 2024 SPLUNK INC.

Scheduled Searches Timechart by Status

Timechart view using trellis for detailed breakdown of scheduled search status over time.

Enable Trellis by host

Scheduled Searches Timechart by Status

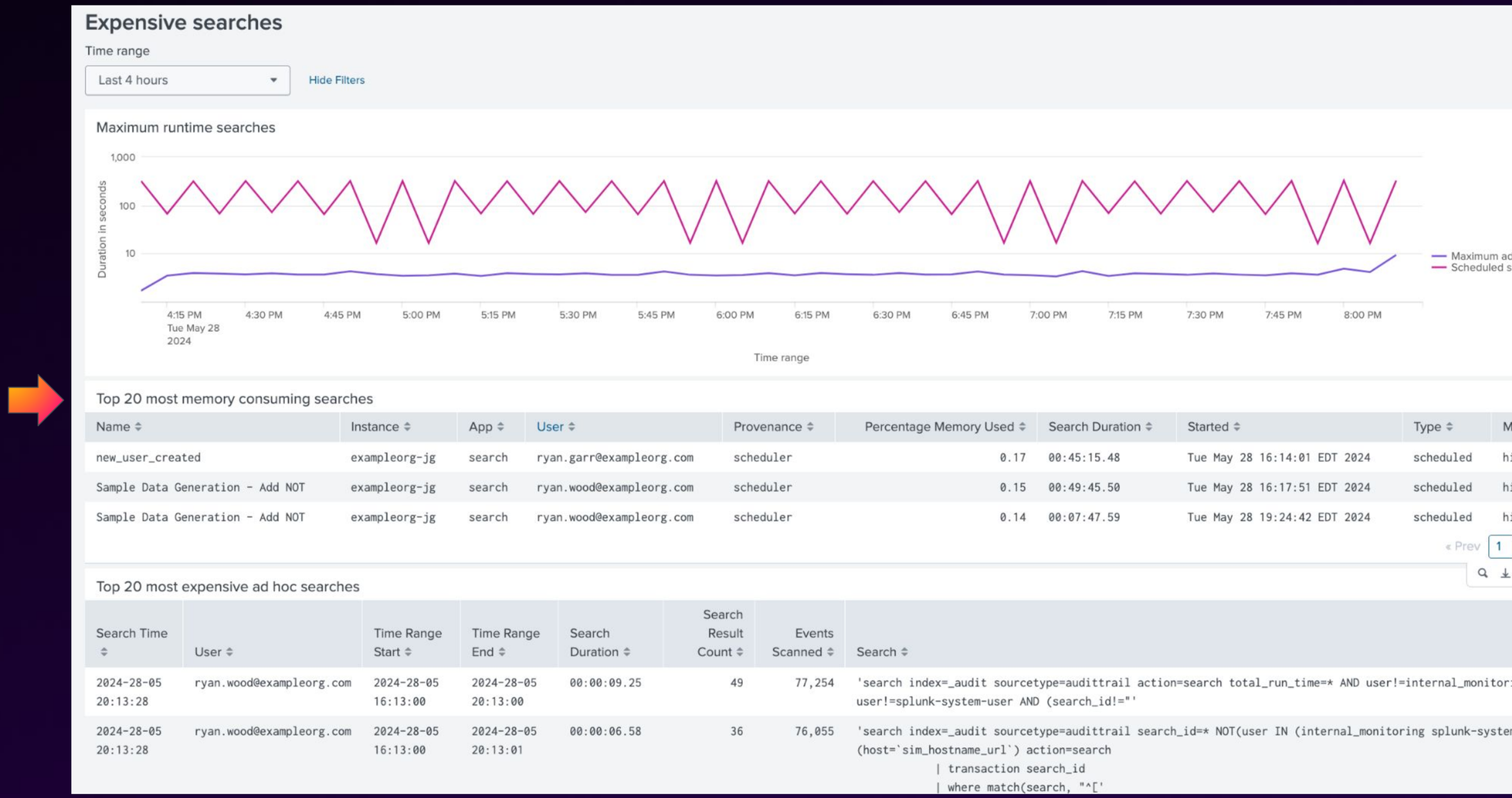
Timechart view showing granular count by status over time by host.
- Enable trellis

```
index=_internal sourcetype=scheduler status=*
| fields _time, savedsearch_id, sid, scheduled_time, status, reason, host
| eval search_id = coalesce(savedsearch_id, sid)
| eval UniqueSearchInstanceIdentifier = search_id."::::".scheduled_time
| eval comment = if( 1==1, null(), "
We treat each unique combination of search_id and scheduled_time as an instance of a search that should execute. We then take the
earliest timestamp for each combination to attribute that status to the timebucket when it began, rather than when it ends or
counting it repeatedly.
This could also be set to take the latest() to attribute the status value to the last event per instance, or not use this logic
at all to take a distinct count in each time bucket regardless of whether that search had already been counted in a previous time
bucket." )
| stats latest(_time) AS _time, latest(*) AS * BY UniqueSearchInstanceIdentifier, status, host
| timechart span=2h dc(eval(if(status="delegated_remote", UniqueSearchInstanceIdentifier, null()))) AS
delegated_remote_dcSearchInstances, dc(eval(if(status="delegated_remote_completion", UniqueSearchInstanceIdentifier, null()))) AS
delegated_remote_completion_dcSearchInstances, dc(eval(if(status="delegated_remote_error", UniqueSearchInstanceIdentifier,
null()))) AS delegated_remote_error_dcSearchInstances, dc(eval(if(status="skipped", UniqueSearchInstanceIdentifier, null()))) AS
skipped_dcSearchInstances, count(eval(if(status="success", UniqueSearchInstanceIdentifier, null()))) AS
success_dcSearchInstances, count(eval(if(status="completed", UniqueSearchInstanceIdentifier, null()))) AS
completed_dcSearchInstances, count(if(eval(status="continued", UniqueSearchInstanceIdentifier, null()))) AS
continued_dcSearchInstances, count(if(eval(status="deferred", UniqueSearchInstanceIdentifier, null()))) AS
deferred_dcSearchInstances BY host
```

Usecase - Identifying Problematic Searches

Cloud Monitoring Console > Search > Expensive Searches

For Splunk Enterprise:
Search Activity: Instance



Usecase - Identifying Problematic Searches

```
_introspection sourcetype=splunk_resource_usage component=PerProcess
```

Report on Highest Memory Consuming searches using...

data.[field]

mem_used - Memory Usage

search_props.app - App context

search_props.label - Savedsearch Name

search_props.sid - Search ID

search_props.user - Who ran the search


```
{ [-]
  component: PerProcess
  data: { [-]
    elapsed: 854.2800
    fd_used: 39
    mem_used: 105.680
    normalized_pct_cpu: 2.15
    page_faults: 0
    pct_cpu: 8.60
    pct_memory: 0.68
    pid: 3728463
    ppid: 64655
    process: splunkd
    process_type: search
    read_mb: 0.000
    search_props: { [-]
      app: search
      delta_scan_count: 0
      label: Sample Data Generation - Add NOT
      mode: historical
      provenance: scheduler
      role: reducer
      scan_count: 0
      search_head: sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com
      sid: remote_sh-i-000e778792a422e91.exampleorg-
jg.splunkcloud.com_prd.ph1_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9
      type: scheduled
      user: ryan.wood@exampleorg.com
```


https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTintrospect#

Usecase - Identifying Problematic Searches

Highest Memory Consuming Searches Using _introspection

```
(index=_introspection sourcetype=splunk_resource_usage data.search_props.sid="*")
| fields data.search_props.sid, data.mem_used, data.search_props.user, data.search_props.app
| rename data.* AS *, search_props.* AS *
``` Gather max value for each SID before normalizing SID and aggregating ```
| stats max(mem_used) AS max_mem_used, values(label) AS savedsearch_label BY sid, user, app
``` Filter to top 10 ```
| sort 10 - max_mem_used
```

sid	user	app	max_mem_used	savedsearch_label
remote_sh-i-000e778792a422e91.guidepoint-jg.splunkcloud.com_prd.ph0_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717179120_35896	ryan.wood@exampleorg.com	search	1023.621	Sample Data Generation - Add NOT
remote_sh-i-000e778792a422e91.guidepoint-jg.splunkcloud.com_prd.ph1_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717179120_35896	ryan.wood@exampleorg.com	search	285.375	Sample Data Generation - Add NOT
scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717179120_35896	ryan.wood@exampleorg.com	search	124.449	Sample Data Generation - Add NOT

What if I wanted to answer...

- Alert actions running in scheduled searches AND the user roles attached to the searches running them?
- What search SPL query was this running?
- Top 10 Memory Consuming Searches AND Execution Time AND SPL Query?

What if I wanted to answer...

- Alert actions running in scheduled searches AND the user roles attached to the searches running them?
- What search SPL query was this running?
- Top 10 Memory Consuming Searches AND Execution Time AND SPL Query?

Search ID!
(SID)

Search - Additional Details

Additional information related to search processes

- tstats-based searches generate introspection data but do not return audittrail or meta information

- Search is a complicated subject with a lot of components, but a good starting place is the Search Manual: https://docs.splunk.com/Documentation/Splunk/latest/Search/GetstartedwithSearch
- Information related to Search Jobs: https://docs.splunk.com/Documentation/SplunkCloud/latest/Search/Aboutjobsandjobmanagement
- On Dispatch directory artifact files, structure, descriptions https://docs.splunk.com/Documentation/SplunkCloud/latesDit/Search/Dispatchdirectoryandsearchartifacts

Search - Optimizing Searches - Resources

Additional information related to optimizing search processes - Resources
Presentations, Utilities, Tools, etc.

- David Paper's Extended Search Reporting Dashboard

Term Efficiency, Search Optimization

- PLA1089C - TSTATS and PREFIX, How to get the most out of your lexicon with walklex, tstats, indexed fields, PREFIX, TERM
- PLA1466B - Fields, Indexed Tokens, and You
- PLA1258C - I Am Speed! Searching on Your Own TERMs With Simple Techniques That 99% Aren't Using!
- TRU1133B - Clara-Fication: More Tstats for Your Buckets

On the topic of understanding job performance:

- TRU1143C - Clara-fication: Job Inspector
- PLA1162B - Clara-Fication: Finding and Improving Expensive Searches

Search - Optimizing Searches - Docs

Additional information related to optimizing search processes - Documentation

- Docs on optimizing searches:
  https://docs.splunk.com/Documentation/Splunk/latest/Search/Aboutoptimization

- Splunk Lantern on Optimizing searches:
  Lantern - Optimizing Search
  Lantern - Writing Better Search Queries

- Job Inspector Information:
  TRU1143C - Clara-fication: Job Inspector
  Docs - Enterprise - View Search Job Properties with Job Inspector
  Docs - Cloud - View Search Job Properties with Job Inspector
  Lantern - Troubleshooting/Investigating Searches with Job Inspector

Search - Important Settings

Settings related to search telemetry

- limits.conf - *track_matching_sourcetypes*
  - Defines whether sourcetype usage is output in audittrail
- limits.conf - *max_audit_sourcetypes*
  - Maximum number of sourcetypes to report on when creating audittrail events
- limits.conf - *record_search_telemetry*
  - Controls whether to record search related metrics in search_telemetry.json in the dispatch dir. It also indexes this file to the _introspection index.
- limits.conf - *subsearch_artifacts_delete_policy*
  - Whether subsearch artifacts are immediately deleted or stay alive for TTL

Default Splunk Indexes with Search Info

Significant default data sources containing SIDs & Search Information.

```
index=_internal
sourcetype=splunkd_remote_searches
```

```
index=_introspection
sourcetype=splunk_resource_usage
component=PerProcess
```

```
index=_audit
sourcetype=audittrail
```

```
index=_internal
sourcetype=scheduler
```

© 2024 SPLUNK INC.

Default Splunk Indexes with Search Info

Significant default data sources containing SIDs & Search Information.

index=_internal
sourcetype=splunkd_remote_searches

Event

05-07-2024 03:59:58.866 +0000 INFO StreamedSearch - Streamed search connection closed: search_id=remote_sh-i-000e778792a422e91.examplehost-jg.splunkcloud.com_prd.ph1_1715054398.80946, se
rver=sh-i-000e778792a422e91.examplehost-jg.splunkcloud.com, active_searches=1, elapsedTime=0.732, search='rdin rdout_sid="prd.ph0_1715054398.80946_sh-i-000e778792a422e91.examplehost-jg.
plunkcloud.com" intermediaries="idx-i-0cf4e337ebc4c2c83.examplehost-jg.splunkcloud.com,idx-i-0e3bd3088c44ea9db.examplehost-jg.splunkcloud.com" order_sensitive="0" allow_partial_results
="0" | tstats summariesonly=false allow_old_summaries=false count WHERE index=_introspection BY splunk_server | prestats count sum(count)', savedsearch_name="", drop_count=0, scan_count
=0, eliminated_buckets=0, considered_events=0, decompressed_slices=0, events_count=0, total_slices=0, considered_buckets=0, search_rawdata_bucketcache_error=0, search_rawdata_bucketcache_
miss=0, search_index_bucketcache_error=0, search_index_bucketcache_hit=0, search_index_bucketcache_miss=0, search_rawdata_bucketcache_hit=0, search_rawdata_bucketcache_miss_wait=0.000, se
arch_index_bucketcache_miss_wait=0.000

host = examplecloudstack.splunkcloud.com | source = /opt/splunk/var/log/splunk/remote_searches.log | sourcetype = splunkd_remote_searches

index=_introspection
sourcetype=splunk_resource_usage
component=PerProcess

{"datetime":"05-07-2024 03:59:57.975 +0000","log_level":"INFO","component":"PerProcess","data":{"pid":"3023884","ppid":"3297339","status":"W","t_count":"11","mem_used":"187.473","pct_mem
ry":"1.20","page_faults":"0","pct_cpu":"106.60","normalized_pct_cpu":"26.65","read_mb":"0.000","written_mb":"6.891","fd_used":"104","elapsed":"2279.7400","process":"splunkd","process_typ
e":"search","workload_pool":"standard_perf","workload_pool_type":"Search","workload_pool_mem_limit":"9218.297","workload_pool_cpu_shares":"358","search_props":{"sid":"remote_sh-i-000e778
92a422e91.examplehost-jg.splunkcloud.com_scheduler_cnlhbi5nYXJyQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD54ae33fa6044677a6_at_1715054100_9739","user":"ryan.garr@examplehostsecurity.co
m","app":"search","label":"new_user_created","provenance":"scheduler","scan_count":"0","delta_scan_count":"0","search_head":"sh-i-000e778792a422e91.examplehost-jg.splunkcloud.com","rol
e":"peer","mode":"historical","type":"scheduled"}}}

Show syntax highlighted

host = examplecloudstack.splunkcloud.com | source = /opt/splunk/var/log/introspection/resource_usage.log | sourcetype = splunk_resource_usage

index=_audit
sourcetype=audittrail

Audit:[timestamp=05-07-2024 03:59:28.488, user=internal_observability, action=search, info=completed, search_id='1715054338.80943', has_error_warn=false, fully_completed_search=true, tota
l_run_time=0.82, event_count=765, result_count=1, available_count=0, scan_count=765, drop_count=0, exec_time=1715054338, api_et=1715054160.000000000, api_lt=1715054220.000000000, api_inde
x_et=N/A, api_index_lt=N/A, search_et=1715054160.000000000, search_lt=1715054220.000000000, is_realtime=0, savedsearch_name="", search_startup_time="64", is_prjob=true, is_flex_search=fa
se, rate_limit_retry_enabled=false, dispatch_artifact_bytes=299008, status_csv_bytes=4096, is_fss3=false, acceleration_id="D347BC42-6954-4328-9ABE-26B97C9DBF62_search_internal_observabil
ty_9e9f02a8f45b44d7", app="search", provenance="N/A", mode="historical_batch", workload_pool=standard_perf, is_proxied=false, searched_buckets=18, eliminated_buckets=0, considered_events=
0, total_slices=0, decompressed_slices=0, duration.command.search.index=0, invocations.command.search.index.bucketcache.hit=18, duration.command.search.index.bucketcache.hit=0, invocatio
s.command.search.index.bucketcache.miss=0, duration.command.search.index.bucketcache.miss=0, invocations.command.search.index.bucketcache.error=0, duration.command.search.rawdata=0, invo
ations.command.search.rawdata.bucketcache.hit=0, duration.command.search.rawdata.bucketcache.hit=0, invocations.command.search.rawdata.bucketcache.miss=0, duration.command.search.rawdata
bucketcache.miss=0, invocations.command.search.rawdata.bucketcache.error=0, roles='observability_role', search='| tstats count where index=_introspection by splunk_server | stats sum(cou
t) AS event_count count AS indexer_count', incomplete_bucket_maps='false', is_federated_search=0, is_fsh_remote_search=0]

host = examplecloudstack.splunkcloud.com | source = audittrail | sourcetype = audittrail

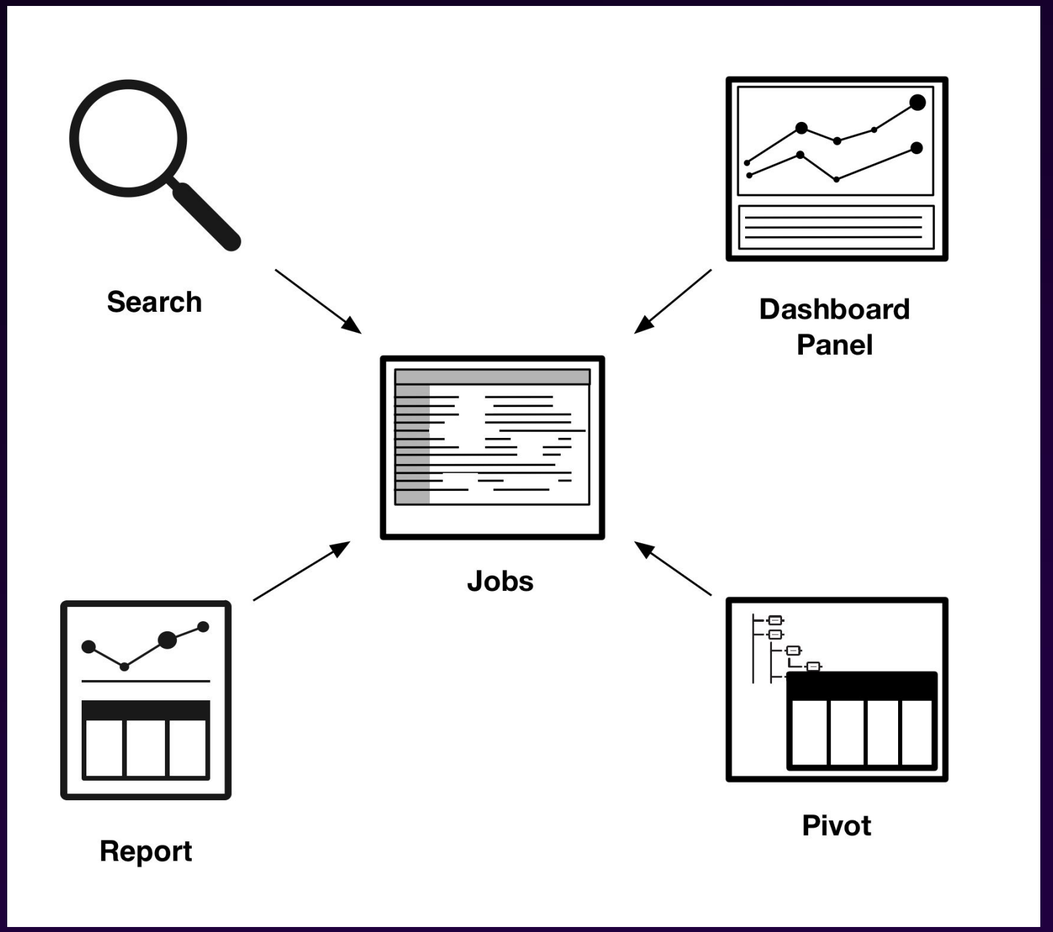
index=_internal
sourcetype=scheduler

05-07-2024 03:59:00.497 +0000 INFO SavedSplunker - savedsearch_id="nobody;skynet-rest;splunk_rest_cluster_status", search_type="scheduled", search_streaming=0, user="admin", app="skynet-
rest", savedsearch_name="splunk_rest_cluster_status", priority=default, status=success, digest_mode=1, durable_cursor=0, scheduled_time=1715054340, window_time=0, dispatch_time=1715054340
0, run_time=0.189, result_count=0, alert_actions="", sid="scheduler__admin_c2t5bmV0LXJlc3Q__RMD56993b9e85281ae60_at_1715054340_20586", suppressed=0, action_time_ms=1, thread_id="AlertNot
fierWorker-0", workload_pool=""

host = examplecloudstack.splunkcloud.com | source = /opt/splunk/var/log/splunk/scheduler.log | sourcetype = scheduler

© 2024 SPLUNK INC.

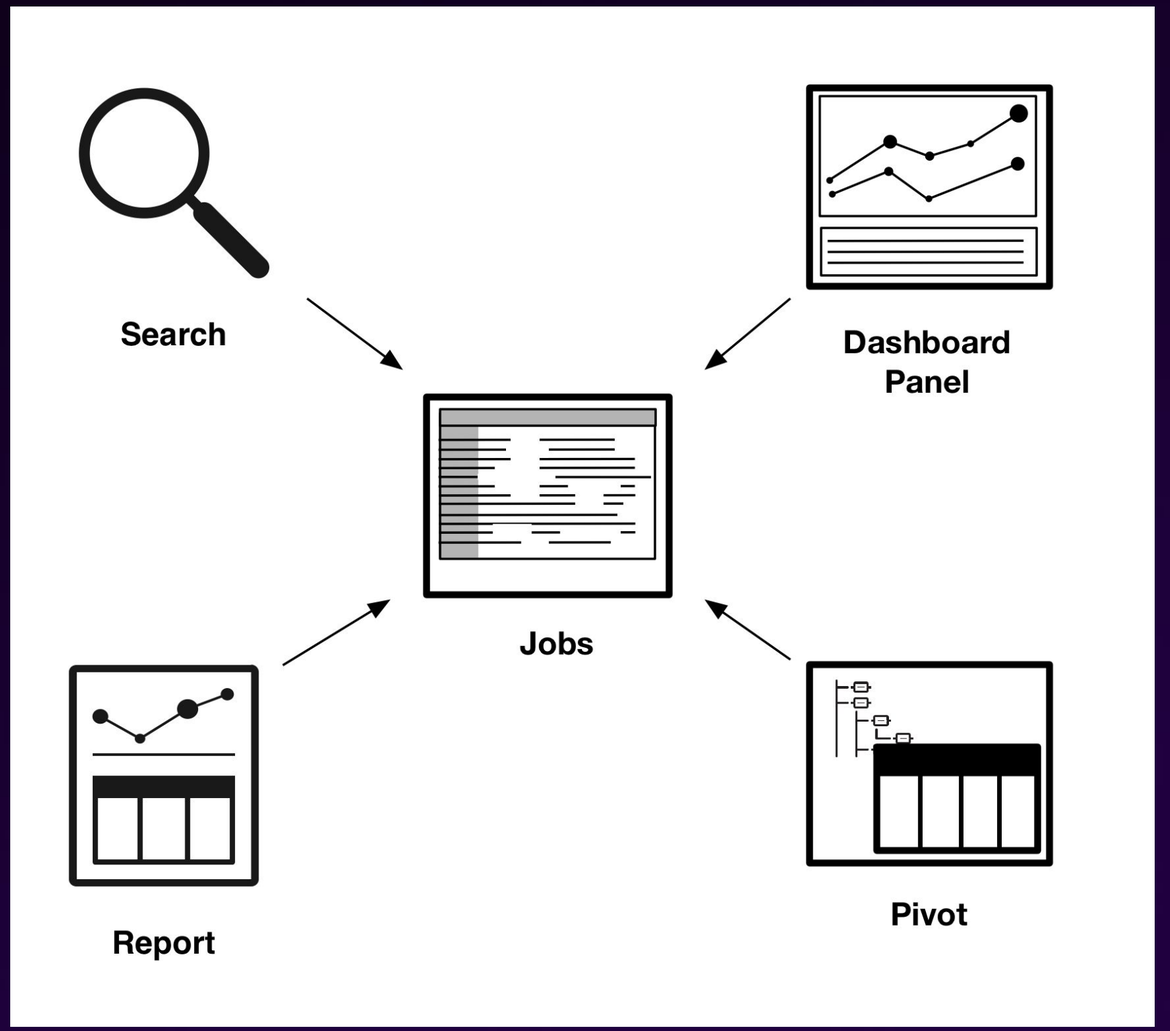
But what *is* a search ID?



Level Set - Search Jobs

A quick refresher

- Each time a search is run, Splunk software creates a ***search job*** in the system.
- Job data is stored within the ***dispatch*** directory of Splunk
- Search job directories are differentiated using unique ID: ***Search ID***.
- **Search ID**s notate the unique directory location of the **artifact files** within the **dispatch** directory



Level Set - Search Types

Types of Searches affect Search ID syntax!

Search Types and Example SID formats available in docs:

Dispatch directory naming conventions

The names of the search-specific directories in the dispatch directory are based on the type of search. For saved and scheduled searches, the name of a search-specific directory is determined by the following conditions.

Type of search	Naming convention	Examples

Saved search	The user requesting the search, the user context the search is run as, the app the search came from, the search string, and the UNIX time.	"count" – run by **admin**, in user context **admin**, saved in app **search**
		`admin__admin__search__count_1347454406.2`
		"Errors in the last 24 hours" – run by **somebody**, in user context **somebody**, saved in app **search**
		`somebody__somebody__search_RMD5473cbac83d6c9db7_1347455134.20`

Scheduled search	The user requesting the search, the user context the search is run as, the app the search came from, the search string, UNIX time, and an internal ID added at the end to avoid name collisions.	"foo" – run by the **scheduler**, with no user context, saved in app **unix**
		`scheduler__nobody__unix__foo_at_1347457380_051d958b8354c580`
		"foo2" - remote peer search on search head **sh01**, with **admin** user context, run by the **scheduler**, saved in app **search**
		`remote_sh01_scheduler__admin__search__foo2_at_1347457920_79152a9a8bf33e5e`

https://docs.splunk.com/Documentation/SplunkCloud/latest/Search/Dispatchdirectoryandsearchartifacts#Dispatch_directory_naming_conventions

Level Set - Search Types

Types of Searches affect Search ID syntax

Types per Docs:

- Local Ad Hoc search
- Saved search
- Scheduled search
- Remote search (peer)
- Real-time search
- Replicated search
- Report acceleration search

Search Types and Example SID formats available in docs:

Dispatch directory naming conventions

The names of the search-specific directories in the dispatch directory are based on the type of search. For saved and scheduled searches, the name of a search-specific directory is determined by the following conditions.

Type of search	Naming convention	Examples

Saved search	The user requesting the search, the user context the search is run as, the app the search came from, the search string, and the UNIX time.	"count" – run by **admin**, in user context **admin**, saved in app **search**
		`admin__admin__search__count_1347454406.2`
		"Errors in the last 24 hours" – run by **somebody**, in user context **somebody**, saved in app **search**
		`somebody__somebody__search_RMD5473cbac83d6c9db7_1347455134.20`

Scheduled search	The user requesting the search, the user context the search is run as, the app the search came from, the search string, UNIX time, and an internal ID added at the end to avoid name collisions.	"foo" – run by the **scheduler**, with no user context, saved in app **unix**
		`scheduler__nobody__unix__foo_at_1347457380_051d958b8354c580`
		"foo2" - remote peer search on search head **sh01**, with **admin** user context, run by the **scheduler**, saved in app **search**
		`remote_sh01_scheduler__admin__search__foo2_at_1347457920_79152a9a8bf33e5e`

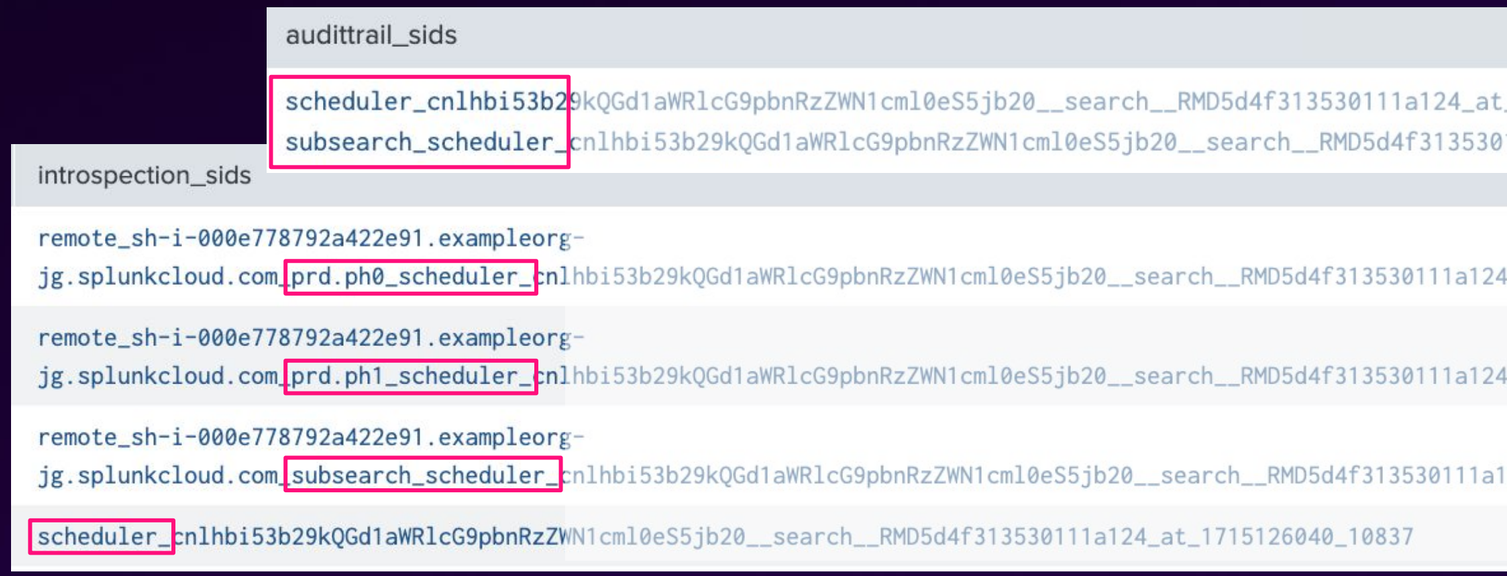
https://docs.splunk.com/Documentation/SplunkCloud/latest/Search/Dispatchdirectoryandsearchartifacts#Dispatch_directory_naming_conventions

Level Set - Search Types

Types of Searches affect Search ID syntax

A few search ID prefixes we need to keep in mind, as they **need** to be normalized to get accurate reporting:

- Scheduled search
  *scheduler_**
- Remote peer
  *remote_**
- Subsearch
  subsearch_*_1347457148.1



Tip: Normalizing SPL available in PDF

https://docs.splunk.com/Documentation/SplunkCloud/latest/Search/Dispatchdirectoryandsearchartifacts#Dispatch_directory_naming_conventions

Usecase - Merging Data Using SIDs

Showing the effect of normalizing SIDs

index=_internal
sourcetype=splunkd_remote_searches

index=_introspection
sourcetype=splunk_resource_usage
component=PerProcess

index=_audit
sourcetype=audittrail

index=_internal
sourcetype=scheduler

```
(index=_audit sourcetype=audittrail search_id="*") OR
(index=_internal sourcetype=scheduler sid="*") OR
(index=_internal sourcetype="splunkd_remote_searches" search_id="*") OR
(index=_introspection sourcetype=splunk_resource_usage data.search_props.sid="*")

| eval unified_sid = case( sourcetype="audittrail", trim(search_id, "'"),
                           sourcetype="scheduler", sid,
                           sourcetype="splunkd_remote_searches", search_id,
                           sourcetype="splunk_resource_usage", 'data.search_props.sid')
| `get_normalized_search_id(unified_sid)`
| rename search_id_normalized AS unified_sid_normalized
| stats dc(unified_sid), dc(unified_sid_normalized) BY index, sourcetype
```

Usecase - Merging Data Using SIDs

Showing the effect of normalizing SIDs

index=_internal
sourcetype=splunkd_remote_searches

index=_introspection
sourcetype=splunk_resource_usage
component=PerProcess

index=_audit
sourcetype=audittrail

index=_internal
sourcetype=scheduler

```
(index=_audit sourcetype=audittrail search_id="*") OR
(index=_internal sourcetype=scheduler sid="*") OR
(index=_internal sourcetype="splunkd_remote_searches" search_id="*") OR
(index=_introspection sourcetype=splunk_resource_usage data.search_props.sid="*")

| eval unified_sid = case( sourcetype="audittrail", trim(search_id, "'"),
                           sourcetype="scheduler", sid,
                           sourcetype="splunkd_remote_searches", search_id,
                           sourcetype="splunk_resource_usage", 'data.search_props.sid')
| `get_normalized_search_id(unified_sid)`
| rename search_id_normalized AS unified_sid_normalized
| stats dc(unified_sid), dc(unified_sid_normalized) BY index, sourcetype
```

index	sourcetype	dc(unified_sid)	dc(unified_sid_normalized)
_introspection	splunk_resource_usage	9,492	6,029
_audit	audittrail	503,110	110,105
_internal	scheduler	66,117	66,116
_internal	splunkd_remote_searches	36,478	21,851

Normalizing SIDs - SPL

SPL for slide aggregating 4 data sources with SID

```
(index=_audit sourcetype=audittrail search_id="*") OR
(index=_internal sourcetype=scheduler sid="*") OR
(index=_internal sourcetype="splunkd_remote_searches" search_id="*") OR
(index=_introspection sourcetype=splunk_resource_usage data.search_props.sid="*")

| eval unified_sid = case( sourcetype="audittrail", trim(search_id, "'"),
                           sourcetype="scheduler", sid,
                           sourcetype="splunkd_remote_searches", search_id,
                           sourcetype="splunk_resource_usage", 'data.search_props.sid')
| `get_normalized_search_id(unified_sid)`
| rename search_id_normalized AS unified_sid_normalized
| stats dc(unified_sid), dc(unified_sid_normalized) BY index, sourcetype
```

© 2024 SPLUNK INC.

Macro: get_normalized_search_id(1)

SPL Gathered from SCMA App (Splunk Cloud Migration Assessment)

SCMA Macro: *get_normalized_search_id(1)* — inserting "unified_sid" field

```
rex field=unified_sid "_(?<search_id_normalized1>\d+[._]\d+)_"
| rex field=unified_sid "(?<search_id_normalized2>\d+[._]\d+$)"
| rex field=unified_sid "(?<search_id_normalized3>^\d+[._]\d+)"
| eval search_id_normalized=if(isnull(search_id_normalized1),search_id_normalized2,search_id_normalized1)
| eval search_id_normalized=if(isnull(search_id_normalized),search_id_normalized3,search_id_normalized)
| eval search_id_normalized=if(isnull(search_id_normalized),search_id,search_id_normalized)
| rex field=search_id_normalized mode=sed "s/\./_/g"
| rex field=search_id_normalized mode=sed "s/^\w+;.*;|^_ACCELERATE_DM_|^_ACCELERATE_|_ACCELERATE_$//g"
| fields - search_id_normalized1,search_id_normalized2,search_id_normalized3
```

What if I wanted to answer...

- Alert actions running in scheduled searches AND the user roles attached to the searches running them?
- What search SPL query was this running?
- Top 10 Memory Consuming Searches AND Execution Time AND SPL Query?

Bring on the `| union`

Usecase - Merging Data Using SIDs

Alert Actions on scheduled searches + User Roles on those searches.

```
_audit audittrail + _internal scheduler
```

© 2024 SPLUNK INC.

Usecase - Merging Data Using SIDs

Alert Actions on scheduled searches + User Roles on those searches.

_audit audittrail + _internal scheduler

- _audit audittrail
  - *search_id*
  - *roles*

Event

```
Audit:[timestamp=05-08-2024 01:16:58.481, user=ryan.wood@exampleorg.com, action=search, info=completed,
search_id='scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887', ha
s_error_warn=false, fully_completed_search=true, total_run_time=156.43, event_count=8872255, [...truncated...], savedse
arch_name="Sample Data Generation", [...truncated...], search_type=scheduled, roles='gps_admin+power+tokens_auth+user',
search='search index=_internal user=*
| stats count by index
| append
    [search index=_audit sourcetype=audittrail
    | stats count by index]', incomplete_bucket_maps='false', is_federated_search=0, is_fsh_remote_search=0]
```

Collapse

host = examplecloudstack.splunkcloud.com | source = audittrail | sourcetype = audittrail

Usecase - Merging Data Using SIDs

Alert Actions on scheduled searches + User Roles on those searches.

_audit audittrail + _internal scheduler

- _audit audittrail
  - *search_id*
  - *roles*

Event

Audit:[timestamp=05-08-2024 01:16:58.481, user=ryan.wood@exampleorg.com, action=search, info=completed, search_id='scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887', has_error_warn=false, fully_completed_search=true, total_run_time=156.43, event_count=8872255, [...truncated...], savedsearch_name="Sample Data Generation", [...truncated...], search_type=scheduled, roles='gps_admin+power+tokens_auth+user', search='search index=_internal user=*
| stats count by index
| append
[search index=_audit sourcetype=audittrail
| stats count by index]', incomplete_bucket_maps='false', is_federated_search=0, is_fsh_remote_search=0]

Collapse

host = examplecloudstack.splunkcloud.com | source = audittrail | sourcetype = audittrail

- _internal scheduler
  - *sid*
  - *alert_actions*

Event

05-08-2024 01:16:36.614 +0000 INFO SavedSplunker - savedsearch_id="nobody;search;Sample Data Generation", search_type="scheduled", search_streaming=0, user="ryan.wood@exampleorg.com", app="search", savedsearch_name="Sample Data Generation", priority=default, status=success, digest_mode=1, durable_cursor=0, scheduled_time=1715130780, window_time=-1, dispatch_time=1715130839, run_time=156.427, result_count=2, alert_actions="email,logevent,lookup", sid="scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887", suppressed=0, action_time_ms=544, thread_id="AlertNotifierWorker-0", workload_pool="standard_perf"

© 2024 SPLUNK INC.

Usecase - Merging Data Using SIDs

Alert Actions on scheduled searches + User Roles on those searches.

`_audit audittrail + _internal scheduler`

- _audit audittrail
  - *search_id*
  - *roles*
- _internal scheduler
  - *sid*
  - *alert_actions*

sourcetype	search_id_normalized	savedsearch_name	alert_actions	roles	user
audittrail	1715130780_10887	Sample Data Generation		gps_admin power tokens_auth user	ryan.wood@exampleorg.com
scheduler	1715130780_10887	Sample Data Generation	email logevent lookup		ryan.wood@exampleorg.com

Usecase - Merging Data Using SIDs

Alert Actions on scheduled searches + User Roles on those searches.

```
_audit audittrail + _internal scheduler
```

search_id_normalized	values(sourcetype)	savedsearch_name	alert_actions	roles	user	app
1715130780_10887	audittrail scheduler	Sample Data Generation	email logevent lookup	gps_admin power tokens_auth user	ryan.wood@exampleorg.com	search

Alert Actions & Roles for Searches

Identifying what alert actions are running and roles attached to those search processes.

Enriched with the SPL Query that ran and the original SIDs before normalization.

```
| union
 [ search (index=_internal sourcetype=scheduler sid="*") alert_actions=* NOT alert_actions=""
 | fields sourcetype, alert_actions, sid, savedsearch_name, user, app
 | eval alert_actions = split(alert_actions, ",")
 | `get_normalized_search_id(sid)`]

 [ search (index=_audit sourcetype=audittrail action=search info=completed search_id="scheduler_*")
 | rex field=_raw ",\ssavedsearch_name=\"(?<savedsearch_name>[^\"]+?)\""
 | rex field=_raw ",\ssearch_id='(?<search_id>[^',]+)"
 | rex field=_raw ",\sapp=\"(?<app>[^\"]+)\""
 | rex field=_raw ",\suser=(?<user>[^,]+)"
 | rex field=_raw ",\sroles=('|\")(?<extracted_roles>(\w+\+)*\w+)('|\"),"
 | rex field=_raw
",\ssearch='(?<searchQuery>[\W\w\n]+?)'((,\sautojoin=)|(\])|(,\sis_federated_search=)|(,\sincomplete_bucket_maps=)|(,\s[^\s=]+=))"
 | eval searchQuery = replace(searchQuery, "',\s[^\s=]+='[^']+$", "")
 | fields sourcetype, search_id, extracted_roles, savedsearch_name, user, app, searchQuery
 | eval extracted_roles = split(extracted_roles, "+")
 | `get_normalized_search_id(search_id)`]
| stats values(savedsearch_name) AS savedsearch_name, values(alert_actions) AS alert_actions, values(extracted_roles) AS roles,
values(user) AS user, values(app) AS app, latest(searchQuery) AS searchQuery,
values(sid) AS scheduler_sids, values(search_id) AS audittrail_sids BY search_id_normalized
```

© 2024 SPLUNK INC.

What if I wanted to answer…

- Alert actions running in scheduled searches AND the user roles attached to the searches running them?
- **What search SPL query was this running?**
- Top 10 Memory Consuming Searches AND Execution Time AND SPL Query?

search_id_normalized	values(sourcetype)	savedsearch_name	alert_actions	roles	user	app	searchQuery
1715130780_10887	audittrail scheduler	Sample Data Generation	email logevent lookup	gps_admin power tokens_auth user	ryan.wood@exampleorg.com	search	search index=_internal user=* | stats count by index | eval foo = "bar" | append [search index=_audit sourcetype=audittrail | stats count by index]

Usecase - Merging Data Using SIDs

Extracting SPL from audittrail

```
| union
 [ search (index=_internal sourcetype=scheduler sid="*") alert_actions=* NOT alert_actions=""
 | fields sourcetype, alert_actions, sid, savedsearch_name, user, app
 | eval alert_actions = split(alert_actions, ",")
 | `get_normalized_search_id(sid)`]

 [ search (index=_audit sourcetype=audittrail action=search info=completed search_id="scheduler_*")
 | rex field=_raw ",\sroles=('|\")(?<extracted_roles>(\w+\+)*\w+)('|\"),"
 | rex field=_raw
",\ssearch='(?<searchQuery>[\W\w\n]+?)'((,\sautojoin=)|(\])|(,\sis_federated_search=)|(,\sincomplete_bucket_maps=)|(,\s[^\s=]+=))"
 | eval searchQuery = replace(searchQuery, "',\s[^\s=]+='[^']+$", "")
 | fields sourcetype, search_id, extracted_roles, savedsearch_name, user, app, searchQuery
 | eval extracted_roles = split(extracted_roles, "+")
 | `get_normalized_search_id(search_id)`]
| stats values(savedsearch_name) AS savedsearch_name, values(alert_actions) AS alert_actions, values(extracted_roles) AS roles,
values(user) AS user, values(app) AS app, latest(searchQuery) AS searchQuery, values(sid) AS scheduler_sids, values(search_id) AS
audittrail_sids BY search_id_normalized
```

^ Alert Actions on scheduled searches + User Roles on those searches.

Usecase - Merging Data Using SIDs

Extracting SPL from audittrail

_audit audittrail

Event

```
Audit:[timestamp=05-08-2024 01:16:58.481, user=ryan.wood@exampleorg.com, action=search, info=completed,
search_id='scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887', has_error_warn=false, fully_c
ompleted_search=true, total_run_time=156.43, event_count=8872255, [...truncated...], savedsearch_name="Sample Data Generation", [...truncated...],
search_type=scheduled, roles='gps_admin+power+tokens_auth+user', search='search index=_internal user=*
| stats count by index
| append
    [search index=_audit sourcetype=audittrail
    | stats count by index]', incomplete_bucket_maps='false', is_federated_search=0, is_fsh_remote_search=0]
```

Collapse

host = examplecloudstack.splunkcloud.com | source = audittrail | sourcetype = audittrail

```
index=_audit sourcetype=audittrail action=search
| rex field=_raw ",\sroles=('|\")(?<extracted_roles>(\w+\+)*\w+)('|\"),"
| rex field=_raw
",\ssearch='(?<searchQuery>[\W\w\n]+?)'((,\sautojoin=)|(\])|(,\sis_federated_search=)|(,\sincomplete_bucket_maps=)|(,\s[^\s=]+=))"
| eval searchQuery = replace(searchQuery, "',\s[^\s=]+='[^']+$", "")
```

Utility SPL Highlight - Regex for Audittrail

Extracting useful elements from audittrail events.

```
(index=_audit sourcetype=audittrail source="*audit.log*" action=search)
| rex field=_raw max_match=0 "sourcetype_count__(?<sourcetype_val>[^=]+)=(?<eventCount>\d+)"
| rex field=_raw ",\ssavedsearch_name=\"(?<savedsearch_name>[^\"]+?)\""
| rex field=_raw ",\ssearch_id='(?<search_id>[^',]+)"
| rex field=_raw ",\sapp=\"(?<app>[^\"]+)\""
| rex field=_raw ",\suser=(?<user>[^,]+)"
| rex field=_raw ",\sinfo=(?<info>[^,]+)"
| rex field=_raw ",\ssearch='(?<searchQuery>[\W\w\n]+?)'((,\sautojoin=)|(\])|(,\sis_federated_search=)|(,\sincomplete_bucket_maps=)|(,\s[^\s=]+=))"
| eval searchQuery = replace(searchQuery, "',\s[^\s=]+='[^']+$", "")
```

Ensures these fields are not pulled from auto kv in SPL queries.

© 2024 SPLUNK INC.

What if I wanted to answer...

- Alert actions running in scheduled searches AND the user roles attached to the searches running them?
- What search SPL query was this running?
- **Top 10 Memory Consuming Searches AND Execution Time AND SPL Query?**

Usecase - Merging Data Using SIDs

Top 10 Memory Consuming Searches with Execution Time & SPL Query

```
_introspection splunk_resource_usage
                    +
_audit audittrail
```

© 2024 SPLUNK INC.

Usecase - Merging Data Using SIDs

Top 10 Memory Consuming Searches with Execution Time & SPL Query

`_introspection splunk_resource_usage`
`+`
`_audit audittrail`

audittrail_sids
scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837
subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837_1715126099.1

introspection_sids
remote_sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com_prd.ph0_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837
remote_sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com_prd.ph1_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837
remote_sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com_subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837_1715126099.1
scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837

Usecase - Merging Data Using SIDs

Top 10 Memory Consuming Searches with Execution Time & SPL Query

_introspection splunk_resource_usage

+

_audit audittrail

```
| `get_normalized_search_id(unified_sid)`
```

** Normalizing SPL available in PDF*

search_id_normalized
1715126040_10837

audittrail_sids
scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837
subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837_1715126099.1

introspection_sids
remote_sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com_prd.ph0_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837
remote_sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com_prd.ph1_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837
remote_sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com_subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837_1715126099.1
scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715126040_10837

Usecase - Merging Data Using SIDs

Top 10 Memory Consuming Searches with Execution Time & SPL Query

_introspection splunk_resource_usage
+
_audit audittrail

```
| `get_normalized_search_id(unified_sid)`
| stats sum(max_mem_used) AS total_mem_used BY search_id_normalized, user, app
| join type=outer search_id_normalized
   [search index=_audit sourcetype=audittrail action=search search_id=*
   ...
```

Tip: Make sure to use audittrail manual extraction patterns from PDF!

exec_time_readable	exec_time	search_id_normalized	savedsearch_name	user	app	total_mem_used	searchQuery
2024-05-07 19:54:59	1715126099	1715126040_10837	Sample Data Generation	ryan.wood@exampleorg.com	search	974.684	search index=_internal user=* | stats count by index | eval foo = "bar" | append [search index=_audit sourcetype=audittrail | stats count by index]

Top 10 Searches by Memory with Exec Time

And SPL Query for SID added to output

```
(index=_introspection sourcetype=splunk_resource_usage data.search_props.sid="*")
| fields data.search_props.sid, data.mem_used, data.search_props.user, data.search_props.app
| rename data.* AS *, search_props.* AS *
``` Gather max value for each SID before normalizing SID and aggregating ```
| stats max(mem_used) AS max_mem_used BY sid, user, app
``` Sum so we get total mem_used including remote SIDs ```
| `get_normalized_search_id(sid)`
| fields - sid
| stats sum(max_mem_used) AS total_mem_used BY search_id_normalized, user, app
``` Filter to top 10 ```
| sort 0 - total_mem_used
``` Add audittrail data, also using normalized SID ```
| join type=outer search_id_normalized
  [search index=_audit sourcetype=audittrail action=search search_id=*
  | rex field=_raw ",\ssearch_id='(?<search_id>[^',]+)"
  | rex field=_raw
",\ssearch='(?<searchQuery>[\W\w\n]+?)'((,\sautojoin=)|(\])|(,\sis_federated_search=)|(,\sincomplete_bucket_maps=)|(,\s[^\s=]+=))"
  | eval searchQuery = replace(searchQuery, "',\s[^\s=]+='[^']+$", "")
  | rex field=_raw ",\ssavedsearch_name=\"(?<savedsearch_name>[^\"]+?)\""
  | fields savedsearch_name, search_id, exec_time, searchQuery
  | `get_normalized_search_id(search_id)`
  | fields - search_id
  | stats latest(*) AS * BY search_id_normalized
  | fields - search_id]
| table exec_time, exec_time_readable, search_id_normalized, savedsearch_name, user, app, total_mem_used, searchQuery
| eval exec_time_readable = strftime(exec_time, "%Y-%m-%d %H:%M:%S")
```

Yes Yes, what about *Data Source Usage?*

Approaches to answering this question.

.conf24

splunk>

Bring on the Data

Data Source Usage by Searches…

Identifying searches using…

- Indexes
- Sourcetypes
- Datamodels
- Lookups
- Eventtypes
- Macros

Data Source Usage by Searches...

during this talk.

Identifying searches using...

- Indexes
- Sourcetypes
- Datamodels
- Lookups
- Eventtypes
- Macros

Which indexes, sourcetypes are my searches using?

Usecase - Data Source Usage

Approached three ways:

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

Regex on SPL

- Using regex to extract index and sourcetype from:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

search.log

- Ingesting search.log files

  OR
- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

Usecase - Data Source Usage

Approached three ways:

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

Challenges:

- Imprecise - Not all search types

Regex on SPL

- Using regex to extract index and sourcetype from:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

Challenges:

- Imprecise due to complexity
- Schema changes over time

search.log

- Ingesting search.log files

  OR
- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

Challenges:

- Imprecise - Point-in-Time
  - Ephemeral Job Artifacts
- Ingest vs Visibility Trade-off

Usecase - Data Source Usage

Approached three ways:

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

Challenges:

- Imprecise - Not all search types

Regex on SPL

- Using regex to extract index and sourcetype from:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

Challenges:

- Imprecise due to complexity
- Schema changes over time

search.log

- Ingesting search.log files

  OR
- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

Challenges:

- Imprecise - Point-in-Time
  - Ephemeral Job Artifacts
- Ingest vs Visibility Trade-off

Data Source Usage

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

Savedsearch in app=splunk_instrumentation:

`instrumentation.usage.search.searchtelemetry.sourcetypeUsage`

`usage.search.searchtelemetry.sourcetypeUsage`	Sourcetype usage.


```
{ [-]
     sourcetypeUsage: [ [-]
       { [-]
         http_event_collector_metrics: 1
         kvstore: 1
         mongod: 3
         search_telemetry: 1
         splunk_disk_objects: 1
         splunk_resource_usage: 1
         splunk_web_service: 3
         splunkd: 11
         splunkd_remote_searches: 3
         splunkd_ui_access: 2
       }
     ]
   }
```


What data is this reporting sourced from?

Data Source Usage

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

Default `audittrail action=search info=completed` events contain sourcetype usage notation:

Event

```
Audit:[timestamp=05-08-2024 01:16:58.481, user=ryan.wood@exampleorg.com, action=search, info=completed, search_id='scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887', has_error_warn=false, fully_completed_search=true, total_run_time=156.43, event_count=8872255, result_count=2, available_count=0, scan_count=131267360, drop_count=0, exec_time=1715130839, api_et=1714449600.000000000, api_lt=1715130480.000000000, api_index_et=N/A, api_index_lt=N/A, search_et=1714449600.000000000, search_lt=1715130480.000000000, is_realtime=0, savedsearch_name="Sample Data Generation", search_startup_time="85", is_prjob=true, is_flex_search=false, rate_limit_retry_enabled=false, dispatch_artifact_bytes=270336, status_csv_bytes=8192, is_fss3=false, acceleration_id="D347BC42-6954-4328-9ABE-26B97C9DBF62_search_ryan.wood@exampleorg.com_06508ace7b9522d3", app="search", provenance="scheduler", mode="historical_batch", workload_pool=standard_perf, is_proxied=false, searched_buckets=45, eliminated_buckets=0, considered_events=131267360, total_slices=365670, decompressed_slices=356464, duration.command.search.index=23852, invocations.command.search.index.bucketcache.hit=45, duration.command.search.index.bucketcache.hit=0, invocations.command.search.index.bucketcache.miss=0, duration.command.search.index.bucketcache.miss=0, invocations.command.search.index.bucketcache.error=0, duration.command.search.rawdata=76906, invocations.command.search.rawdata.bucketcache.hit=27, duration.command.search.rawdata.bucketcache.hit=0, invocations.command.search.rawdata.bucketcache.miss=0, duration.command.search.rawdata.bucketcache.miss=0, invocations.command.search.rawdata.bucketcache.error=0,
sourcetype_count__scheduler=66950, sourcetype_count__secure_gateway_app_internal_log=1962, sourcetype_count__splunk_audit=317, sourcetype_count__splunk_python=1427, sourcetype_count__splunk_secure_gateway_modular_input.log=476, sourcetype_count__splunk_secure_gateway_modular_input.log-too_small=183, sourcetype_count__splunk_web_access=47722, sourcetype_count__splunk_web_service=1424, sourcetype_count__splunkd=34076, sourcetype_count__splunkd_access=8233885, sourcetype_count__splunkd_ui_access=481442, search_type=scheduled, roles='gps_admin+power+tokens_auth+user', search='search index=_internal user=*
| stats count by index
| append
    [search index=_audit sourcetype=audittrail
    | stats count by index]', incomplete_bucket_maps='false', is_federated_search=0, is_fsh_remote_search=0]
```

Collapse

host = examplecloudstack.splunkcloud.com | source = audittrail | sourcetype = audittrail

Data Source Usage

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

```
index=_audit
sourcetype=audittrail
action=search info=completed
"sourcetype_count__*"
```

Default `audittrail action=search info=completed` events contain sourcetype usage notation:

```
| rex field=_raw max_match=0 "sourcetype_count__(?<sourcetype_val>[^=]+)=(?<eventCount>\d+)"
```

Data Source Usage

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

```
index=_audit
sourcetype=audittrail
action=search info=completed
"sourcetype_count__*"
```

Default `audittrail action=search info=completed` events contain sourcetype usage notation:

```
| rex field=_raw max_match=0 "sourcetype_count__(?<sourcetype_val>[^=]+)=(?<eventCount>\d+)"
```

```
index=_audit sourcetype=audittrail action=search info=completed
"*sourcetype_count__*"
| rex field=_raw max_match=0 "sourcetype_count__(?<sourcetype_val>\w+)=(?<eventCount>\d+)"
| rex field=_raw ",\ssavedsearch_name=\"(?<savedsearch_name>[^\"]+?)\""
| eval savedsearch_name = if(savedsearch_name="" OR isnull(savedsearch_name), "None", savedsearch_name)
| rex field=_raw ",\sapp=\"(?<app>[^\"]+)\""
| rex field=_raw ",\suser=(?<user>[^,]+)"
| stats sum(eventCount) AS total_eventCount, values(user) AS user_values, values(app) AS app_values,
values(provenance) AS provenance_values, dc(search_id) AS dc_searches BY savedsearch_name, sourcetype_val
```

Data Source Usage

Instrumentation

➢ sourcetype=audittrail

sourcetype_count__<sourcetype>

```
index=_audit
sourcetype=audittrail
action=search info=completed
"sourcetype_count__*"
```

Default `audittrail action=search info=completed` events contain sourcetype usage notation:

```
| rex field=_raw max_match=0 "sourcetype_count__(?<sourcetype_val>[^=]+)=(?<eventCount>\d+)"
```

```
index=_audit sourcetype=audittrail action=search info=completed
"*sourcetype_count__*"
| rex field=_raw max_match=0 "sourcetype_count__(?<sourcetype_val>\w+)=(?<eventCount>\d+)"
| rex field=_raw ",\ssavedsearch_name=\"(?<savedsearch_name>[^\"]+?)\""
| eval savedsearch_name = if(savedsearch_name="" OR isnull(savedsearch_name), "None", savedsearch_name)
| rex field=_raw ",\sapp=\"(?<app>[^\"]+)\""
| rex field=_raw ",\suser=(?<user>[^,]+)"
| stats sum(eventCount) AS total_eventCount, values(user) AS user_values, values(app) AS app_values,
values(provenance) AS provenance_values, dc(search_id) AS dc_searches BY savedsearch_name, sourcetype_val
```

savedsearch_name	sourcetype_val	total_eventCount	user_values	app_values	provenance_values
None	search_log_events	2395727424	ryan.wood@exampleorg.com	search	N/A UI:Search
Sample Data Generation	splunk_python	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunk_web_service	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunkd_access	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	scheduler	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunk_web_access	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunkd	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunkd_ui_access	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	secure_gateway_app_internal_log	225360056	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunk_audit	109909761	ryan.wood@exampleorg.com	search	scheduler

Data Source Usage

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

```
index=_audit
sourcetype=audittrail
action=search info=completed
"sourcetype_count__*"
```

Default `audittrail action=search info=completed` events contain sourcetype usage notation:

SPL text block that can be copied:

```
index=_audit sourcetype=audittrail action=search info=completed
"*sourcetype_count__*"
| rex field=_raw max_match=0 "sourcetype_count__(?<sourcetype_val>[^=]+)=(?<eventCount>\d+)"
| rex field=_raw ",\ssavedsearch_name=\"(?<savedsearch_name>[^\"]+?)\""
| eval savedsearch_name = if(savedsearch_name="" OR isnull(savedsearch_name), "None", savedsearch_name)
| rex field=_raw ",\sapp=\"(?<app>[^\"]+)\""
| rex field=_raw ",\suser=(?<user>[^,]+)"
| rex field=_raw ",\ssearch_id='(?<search_id>[^',]+)"
| stats sum(eventCount) AS total_eventCount, values(user) AS user_values, values(app) AS app_values,
   values(provenance) AS provenance_values, dc(search_id) AS dc_searches BY savedsearch_name, sourcetype_val
```

savedsearch_name	sourcetype_val	total_eventCount	user_values	app_values	provenance_values
None	search_log_events	2395727424	ryan.wood@exampleorg.com	search	N/A UI:Search
Sample Data Generation	splunk_python	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunk_web_service	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunkd_access	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	scheduler	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunk_web_access	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunkd	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunkd_ui_access	225387831	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	secure_gateway_app_internal_log	225360056	ryan.wood@exampleorg.com	search	scheduler
Sample Data Generation	splunk_audit	109909761	ryan.wood@exampleorg.com	search	scheduler

Usecase - Data Source Usage

Approached three ways:

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

Challenges:

- Imprecise - Not all search types

Regex on SPL

- Using regex to extract index and sourcetype from:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

Challenges:

- Imprecise due to complexity
- Schema changes over time

search.log

- Ingesting search.log files

  OR
- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

Challenges:

- Imprecise - Point-in-Time
  - Ephemeral Job Artifacts
- Ingest vs Visibility Trade-off

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

index=_audit sourcetype=audittrail action=search

Event

Audit:[timestamp=05-08-2024 01:16:58.481, user=ryan.wood@exampleorg.com, action=search, info=completed, search_id='scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887', has_error_war ompleted_search=true, total_run_time=156.43, event_count=8872255, [...truncated...], savedsearch_name="Sample Data Generation", [search_type=scheduled, roles='gps_admin+power+tokens_auth+user', search='search index=_internal user=* | stats count by index | append [search index=_audit sourcetype=audittrail | stats count by index]', incomplete_bucket_maps='false', is_federated_search=0, is_fsh_remote_search=0]

Collapse

host = examplecloudstack.splunkcloud.com | source = audittrail | sourcetype = audittrail

index=_internal sourcetype=splunkd_remote_searches

Event

05-08-2024 01:16:36.026 +0000 INFO StreamedSearch - Streamed search connection closed: search_id=remote_sh-i-000e778792a422e91.exampleorg prd.ph0_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887, server=sh-i-000e778792a42 plunkcloud.com, active_searches=1, elapsedTime=151.465, search='litsearch (index=_internal ((sourcetype=DhcpSrvLog msdhcp_user="*") OR (sourcetype="WMI:UserAccounts" Name="*") OR (sourcetype="WM ation" User="*") OR (sourcetype=ps USER="*") OR (sourcetype=top USER="*") OR user="*" OR (sourcetype=audittrail (uid="*" OR user_id="*" OR etype=linux_audit OR sourcetype="o365:cas:api" OR sourcetype="o365:management:activity" OR source="WinEventLog:Security" OR source="WinEve rce="XmlWinEventLog:Security")) | litsearch (index=_internal user=*) | addinfo type=count label=prereport_events track_fieldmeta_events=t lorder=t "index" "prestats_reserved_*" "psrsvd_*" | prestats count by index | rdout partition_method="hash" num_of_intermediaries="2" ha r_sensitive="0"', savedsearch_name="Sample Data Generation", drop_count=0, scan_count=43648633, eliminated_buckets=0, considered_events=43 _slices=118727, events_count=43648633, total_slices=121260, considered_buckets=14, search_rawdata_bucketcache_error=0, search_rawdata_buck ch_index_bucketcache_error=0, search_index_bucketcache_hit=14, search_index_bucketcache_miss=0, search_rawdata_bucketcache_hit=8, search_r iss_wait=0.000, search_index_bucketcache_miss_wait=0.000

host = examplecloudstack.splunkcloud.com | source = /opt/splunk/var/log/splunk/remote_searches.log | sourcetype = splunkd_remote_searches

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

What is all of this extra SPL?

- *litsearch* is the literal search run on peers
- When the query run doesn't include *source=* or *sourcetype=*, global eventtype conditions are applied.
  - **This can have significant impact to search efficiency**

index=_audit sourcetype=audittrail action=search

Event

Audit:[timestamp=05-08-2024 01:16:58.481, user=ryan.wood@exampleorg.com, action=search, info=completed, search_id='scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887', has_error_war ompleted_search=true, total_run_time=156.43, event_count=8872255, [...truncated...], savedsearch_name="Sample Data Generation", [search_type=scheduled, roles='gps_admin+power+tokens_auth+user', search='search index=_internal user=* | stats count by index | append [search index=_audit sourcetype=audittrail | stats count by index]', incomplete_bucket_maps='false', is_federated_search=0, is_fsh_remote_search=0]

Collapse

host = examplecloudstack.splunkcloud.com | source = audittrail | sourcetype = audittrail

index=_internal sourcetype=splunkd_remote_searches

Event

05-08-2024 01:16:36.026 +0000 INFO StreamedSearch - Streamed search connection closed: search_id=remote_sh-i-000e778792a422e9 mpleorg prd.ph0_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5d4f313530111a124_at_1715130780_10887, server=sh 778792a42 lunkcloud.com, active_searches=1, elapsedTime=151.465, search='litsearch (index=_internal ((sourcetype=DhcpSrvLog msdhcp_user="*") OR (sourcetype="WMI:UserAccounts" Na (sourcetype="WM ation" User="*") OR (sourcetype=ps USER="*") OR (sourcetype=top USER="*") OR user="*" OR (sourcetype=audittrail (us *" OR user_id="*" OR etype=linux_audit OR sourcetype="o365:cas:api" OR sourcetype="o365:management:activity" OR source="WinEventLog:Security" OR source="WinEve rce="XmlWinEventLog:Security")) | litsearch (index=_internal user=*) | addinfo type=count label=prereport_events track_fieldmeta_events=t lorder=t "index" "prestats_reserved_*" "psrsvd_*" | prestats count by index | rdout partition_method="hash" num_of_intermediaries="2" ha r_sensitive="0"', savedsearch_name="Sample Data Generation", drop_count=0, scan_count=43648633, eliminated_buckets=0, considered_events=43 _slices=118727, events_count=43648633, total_slices=121260, considered_buckets=14, search_rawdata_bucketcache_error=0, search_rawdata_buck ch_index_bucketcache_error=0, search_index_bucketcache_hit=14, search_index_bucketcache_miss=0, search_rawdata_bucketcache_hit=8, search_r iss_wait=0.000, search_index_bucketcache_miss_wait=0.000

host = examplecloudstack.splunkcloud.com | source = /opt/splunk/var/log/splunk/remote_searches.log | sourcetype = splunkd_remote_searches

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

Challenges of audittrail/remote_searches.log:

- SPL can be any string - requires complex regex patterns
- Handling wildcard references
- Abstraction - Macros, Eventtypes, RBAC on Indexes
- Multiple formats for specifying index/sourcetype
- Subsearches
- Unexpected bits!
  - ie. Remote searches containing unrelated eventtype SPL

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

Challenges of audittrail/remote_searches.log:

- SPL can be any string - requires complex regex patterns
- Handling wildcard references
- Abstraction - Macros, Eventtypes, RBAC on Indexes
- Multiple formats for specifying index/sourcetype
- Subsearches
- Unexpected bits!

IndexerLevel - RemoteSearches Indexes Stats App: Alerts for Splunk Admins Save Save As View Crea

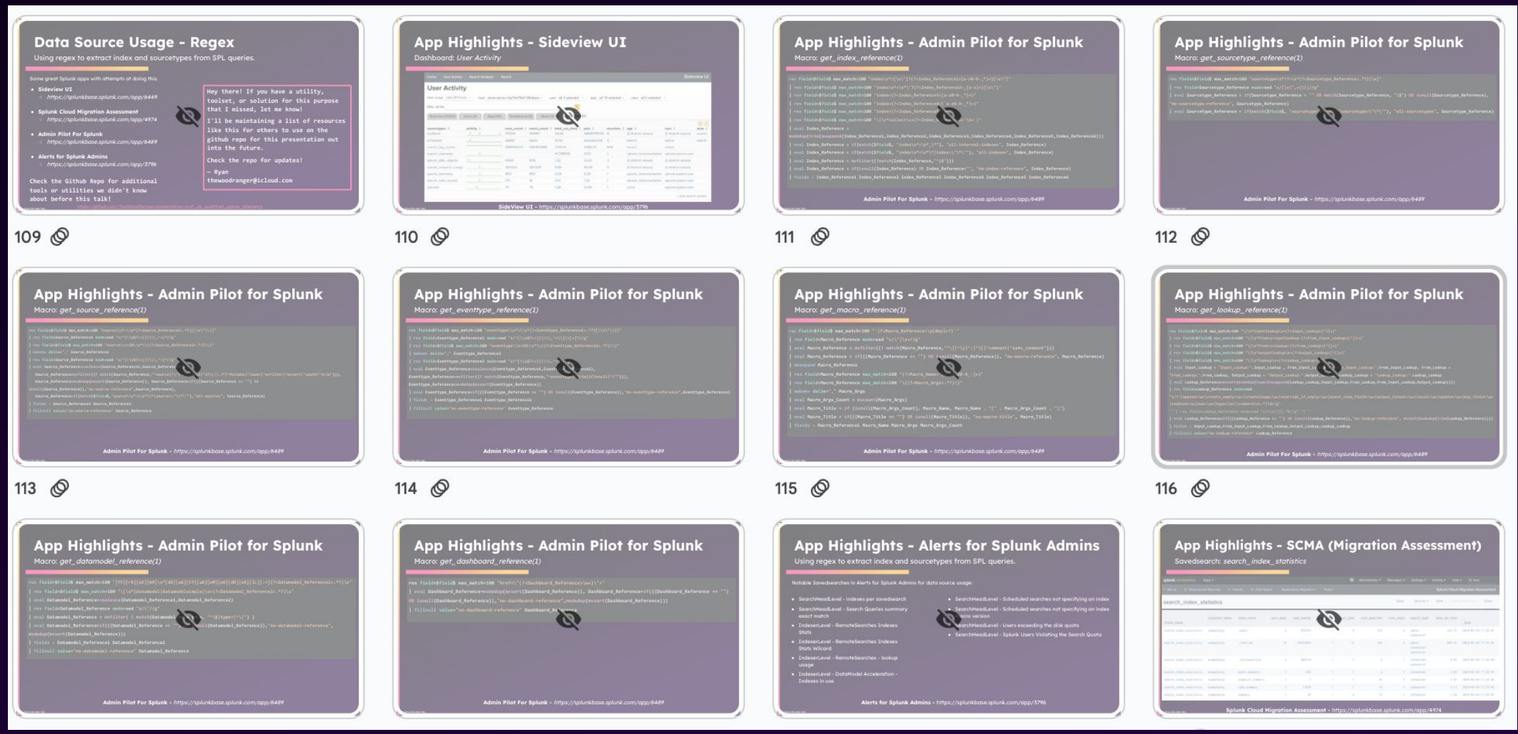
```
    index=_internal sourcetype=splunkd_remote_searches source="/opt/splunk/var/log/splunk/remote_searches.log" terminated: OR closed:
| rex "(?s) elapsedTime=(?P<elapsedTime>[0-9\.]+), search='(?P<search>.*?)(', savedsearch_name|\", drop_count=\d+)"
| regex search!="^(pretypeahead|copybuckets)"
| rex "drop_count=[0-9]+, scan_count=(?P<scan_count>[0-9]+)"
| rex "total_slices=[0-9]+, considered_buckets=(?P<considered_count>[0-9]+)"
| rex "(,|}\.\.\.) savedsearch_name=\"(?P<savedsearch_name>[^\"]*)\","
| rex "(terminated|closed): search_id=(?P<search_id>[^,]+)"
| regex search="^(litsearch|mcatalog|mstats|mlitsearch|litmstats|tstats|presummarize)"
| rex field=search max_match=50 "(?s)\|?\s*(mlitsearch)\s+.*?\[(?P<subsearch>.*?)\]\s*(\||$)"
| rex field=search "(?s)(?P<prepipe>\s*\|?([^\|]+))"
| nomv subsearch
| eval subsearch=if(isnull(subsearch),"",subsearch)
| eval prepipe = prepipe . " " . subsearch
| eval search=prepipe
| search `comment("The (index=* OR index=_*) in                    ndex> is a common use case for enterprise security, also some individuals like doing a similar trick so remove the index=*... as this is not a
    wildcard index search")`
| rex field=search "(?P<esstylewildcard>\(\s*                    ndex=_\*\s*\))"
| rex mode=sed field=search "s/search index=\s*                    /search index=/"
| search `comment("Extract out index= or index                    avoid NOT index in (...) and NOT index=... and also NOT (...anything) statements")`
| rex field=search "(?s)(NOT\s+index(\s*=\                    ]+\))|(index(\s*=\s*|::)(?P<indexregex>[\*A-Za-z0-9-_]+))" max_match=50
| rex field=search "(?s)(NOT\s+index(\s                    ))|(index(\s*=\s*|::)\"?(?P<indexregex2>[\*A-Za-z0-9-_]+))" max_match=50
| rex field=search "\s+(?P<skipping>\                    ng 46464 bytes}..., then drop the last index found in the regex as it is likely invalid")`
| search `comment("If skipping is in
| eval indexregex=if(isnull(skipp                    ex)
| eval indexregex2=if(i                    egex2)
| eval indexes=mvappend(                    dexes)
| eval indexes=if(isnotnu
| eval multi=if(mvcount(mvde
| rex field=search_id "^remote_(?P<sid>.*)
| rex "search_id=[^,]+,\s+server=(?P<serv
| eval server_with_underscore = server.
| eval sid=replace(sid, server_with_und
```

Data Source Usage - Regex

Using regex to extract index and sourcetypes from SPL queries.

Some great Splunk apps we ran across attempting to do this:

- **Sideview UI**
  - Dashboard:
    User Activity
- **Splunk Cloud Migration Assessment**
  - Savedsearch:
    search_index_statistics
- **Admin Pilot For Splunk**
  - Macros:
    get_index_reference(1),
    get_sourcetype_reference(1)
- **Alerts for Splunk Admins**
  - Savedsearches:
    IndexerLevel - RemoteSearches Indexes Stats
    SearchHeadLevel - indexes per savedsearch



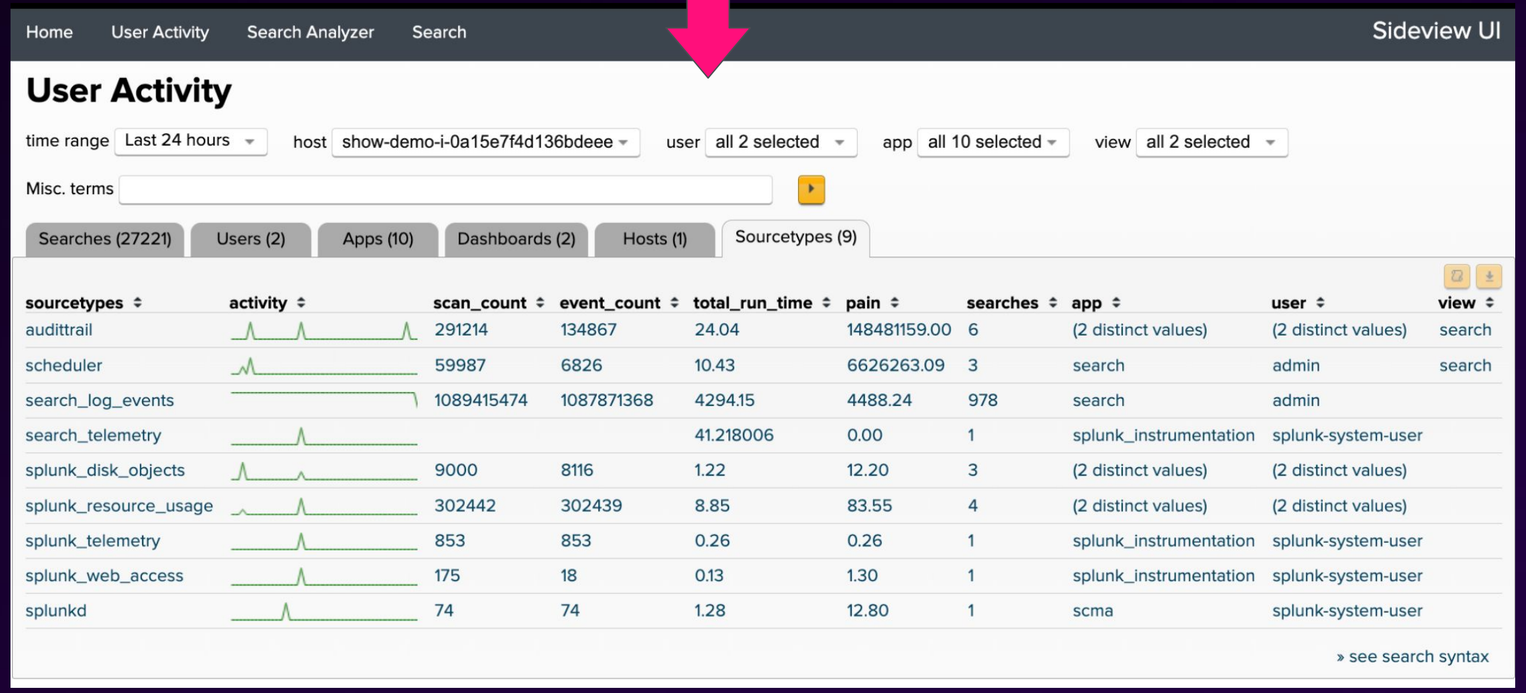
... And Much More in the PDF!

Data Source Usage - Regex

Using regex to extract index and sourcetypes from SPL queries.

Some great Splunk apps we ran across attempting to do this:

- **Sideview UI**
  - Dashboard:
    User Activity



Data Source Usage - Regex

Using regex to extract index and sourcetypes from SPL queries.

Some great Splunk apps with attempts at doing this:

- **Sideview UI**
  - *https://splunkbase.splunk.com/app/6449*
- **Splunk Cloud Migration Assessment**
  - *https://splunkbase.splunk.com/app/4974*
- **Admin Pilot For Splunk**
  - *https://splunkbase.splunk.com/app/6489*
- **Alerts for Splunk Admins**
  - *https://splunkbase.splunk.com/app/3796*

Check the Github Repo for additional tools or utilities we didn't know about before this talk!

Hey there! If you have a utility, toolset, or solution for this purpose that I missed, let me know!

I'll be maintaining a list of resources like this for others to use on the github repo for this presentation out into the future.

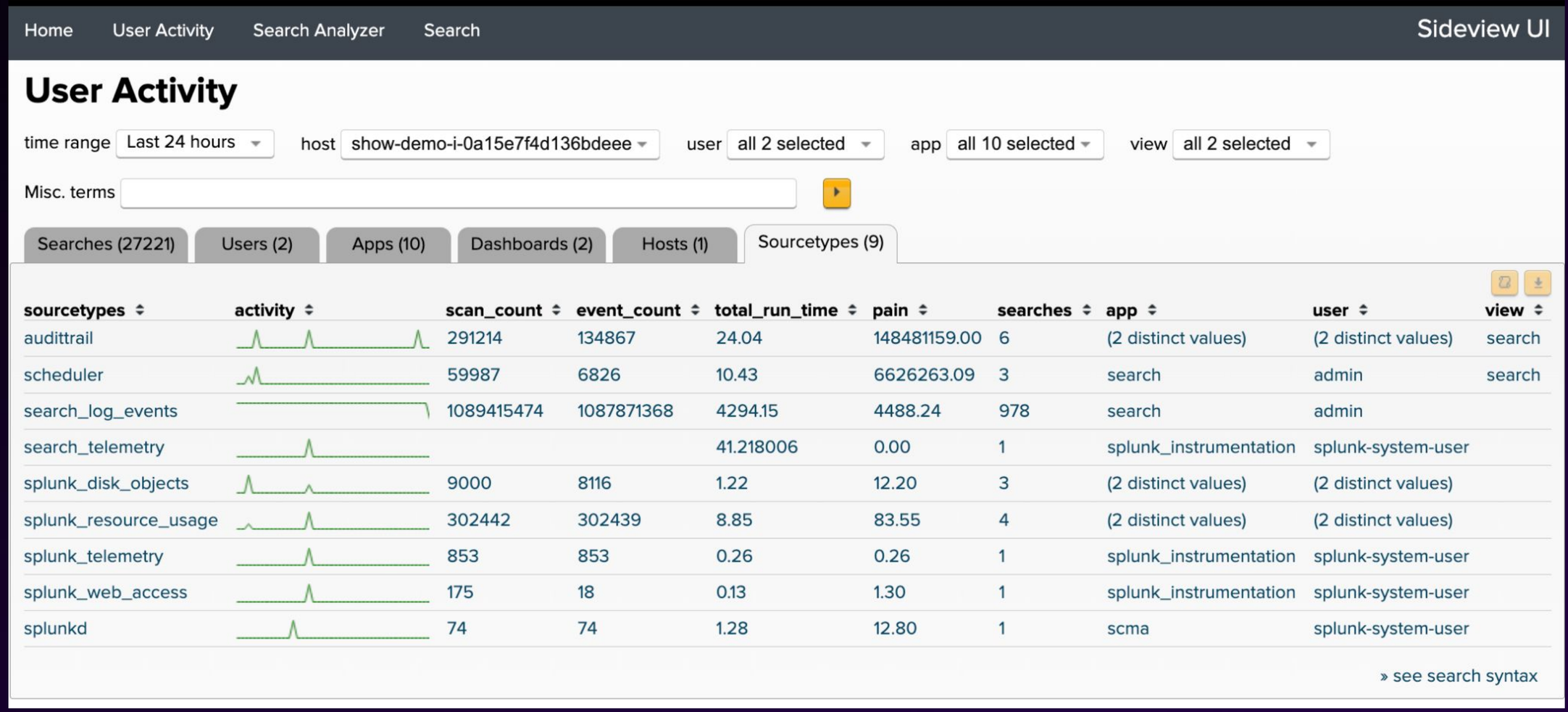
Check the repo for updates!

– Ryan
thewoodranger@icloud.com

https://github.com/TheWoodRanger/presentation-conf_24_audittrail_native_telemetry

App Highlights - Sideview UI

Dashboard: *User Activity*



sourcetypes	activity	scan_count	event_count	total_run_time	pain	searches	app	user	view
audittrail		291214	134867	24.04	148481159.00	6	(2 distinct values)	(2 distinct values)	search
scheduler		59987	6826	10.43	6626263.09	3	search	admin	search
search_log_events		1089415474	1087871368	4294.15	4488.24	978	search	admin	
search_telemetry				41.218006	0.00	1	splunk_instrumentation	splunk-system-user	
splunk_disk_objects		9000	8116	1.22	12.20	3	(2 distinct values)	(2 distinct values)	
splunk_resource_usage		302442	302439	8.85	83.55	4	(2 distinct values)	(2 distinct values)	
splunk_telemetry		853	853	0.26	0.26	1	splunk_instrumentation	splunk-system-user	
splunk_web_access		175	18	0.13	1.30	1	splunk_instrumentation	splunk-system-user	
splunkd		74	74	1.28	12.80	1	scma	splunk-system-user	

SideView UI - https://splunkbase.splunk.com/app/3796

App Highlights - Admin Pilot for Splunk

Macro: *get_index_reference(1)*

```
rex field=$field$ max_match=100 "index\s*=[\s\"]?(?<Index_Reference1>[a-z0-9-_*]+)[\s\"]"
| rex field=$field$ max_match=100 "index\s*=\s*\"?(?<Index_Reference2>_[a-z]+)[\s\"]"
| rex field=$field$ max_match=100 "index=(?<Index_Reference3>[a-z0-9-_*]+)"
| rex field=$field$ max_match=100 "index=(?<Index_Reference4>[`a-z0-9-_*]+)"
| rex field=$field$ max_match=100 "index=(?<Index_Reference5>\w+)"
| rex field=$field$ max_match=100 "\|\s*collect\s+(?<Index_Reference6>`\S+`)"
| eval Index_Reference =
mvdedup(trim(mvappend(Index_Reference1,Index_Reference2,Index_Reference3,Index_Reference4,Index_Reference5,Index_Reference6)))
| eval Index_Reference = if(match($field$, "index\s*=\s*_\*"), "all-internal-indexes", Index_Reference)
| eval Index_Reference = if(match($field$, "index\s*=\s*\*|index=\"\*\""), "all-indexes", Index_Reference)
| eval Index_Reference = mvfilter((!match(Index_Reference,"^1$")))
| eval Index_Reference = if(isnull(Index_Reference) OR Index_Reference="", "no-index-reference", Index_Reference)
| fields - Index_Reference1 Index_Reference2 Index_Reference3 Index_Reference4 Index_Reference5 Index_Reference6
```

Admin Pilot For Splunk - *https://splunkbase.splunk.com/app/6489*

App Highlights - Admin Pilot for Splunk

Macro: *get_sourcetype_reference(1)*

```
rex field=$field$ max_match=100 "sourcetype\s*!?=\s*(?<Sourcetype_Reference>.*?)[\s]"
| rex field=Sourcetype_Reference mode=sed "s/[\s\",=()|]//g"
| eval Sourcetype_Reference = if(Sourcetype_Reference = "" OR match(Sourcetype_Reference, "\$") OR isnull(Sourcetype_Reference), "no-sourcetype-reference", Sourcetype_Reference)
| eval Sourcetype_Reference = if(match($field$, "sourcetype\s*=\s*\*|sourcetype=\"\*\""), "all-sourcetypes", Sourcetype_Reference)
```

Admin Pilot For Splunk - *https://splunkbase.splunk.com/app/6489*

App Highlights - Admin Pilot for Splunk

Macro: *get_source_reference(1)*

```
rex field=$field$ max_match=100 "source\\s*=\\s*(?<Source_Reference1>.*?)[\\s\"\\|]"
| rex field=Source_Reference1 mode=sed "s/^[\\s$?><()\\\\,^=]*//g"
| rex field=$field$ max_match=100 "source\\s+IN\\s*\\((?<Source_Reference2>.*?)\\)"
| makemv delim="," Source_Reference2
| rex field=Source_Reference2 mode=sed "s/^[\\s$?><()\\\\,^=]*//g"
| eval Source_Reference=coalesce(Source_Reference1,Source_Reference2),
   Source_Reference=mvfilter((! match(Source_Reference,"^source|^\"|^ifisnull|^if\(|\.\*|^Mvindex|^lower|^mvfilter|^mvsort|^spath|^trim"))),
   Source_Reference=mvdedup(mvsort(Source_Reference)), Source_Reference=if(((Source_Reference == "") OR
isnull(Source_Reference)),"no-source-reference",Source_Reference),
   Source_Reference=if(match($field$,"source\\s*=\\s*\\*|source=\"\\*\""),"all-sources", Source_Reference)
| fields - Source_Reference1 Source_Reference2
| fillnull value="no-source-reference" Source_Reference
```

Admin Pilot For Splunk - *https://splunkbase.splunk.com/app/6489*

App Highlights - Admin Pilot for Splunk

Macro: *get_eventtype_reference(1)*

```
rex field=$field$ max_match=100 "eventtype\\s*=\\s*(?<Eventtype_Reference1>.*?)[\\s\"\\|]"
| rex field=Eventtype_Reference1 mode=sed "s/^[\\s$?><()\\\\,^=\\]\\[+]*//g"
| rex field=$field$ max_match=100 "eventtype\\s+IN\\s*\\((?<Eventtype_Reference2>.*?)\\)"
| makemv delim="," Eventtype_Reference2
| rex field=Eventtype_Reference2 mode=sed "s/^[\\s$?><()\\\\,^=]*//g"
| eval Eventtype_Reference=coalesce(Eventtype_Reference1,Eventtype_Reference2),
Eventtype_Reference=mvfilter((! match(Eventtype_Reference,"^eventtype|^trim|ifisnull|^\""))),
Eventtype_Reference=mvdedup(mvsort(Eventtype_Reference))
| eval Eventtype_Reference=if(((Eventtype_Reference == "") OR isnull(Eventtype_Reference)),"no-eventtype-reference",Eventtype_Reference)
| fields - Eventtype_Reference1 Eventtype_Reference2
| fillnull value="no-eventtype-reference" Eventtype_Reference
```

Admin Pilot For Splunk - *https://splunkbase.splunk.com/app/6489*

App Highlights - Admin Pilot for Splunk

Macro: *get_macro_reference(1)*

```
rex field=$field$ max_match=100 "`(?<Macro_Reference>\p{Any}+?)`"
| rex field=Macro_Reference mode=sed "s/\"|\s+//g"
| eval Macro_Reference = mvfilter((! match(Macro_Reference,"^\||^\)|^:|^\[|^comment|^ia4s_comment")))
| eval Macro_Reference = if(((Macro_Reference == "") OR isnull(Macro_Reference)), "no-macro-reference", Macro_Reference)
| mvexpand Macro_Reference
| rex field=Macro_Reference max_match=100 "(?<Macro_Name>^[a-zA-Z0-9_-]+)"
| rex field=Macro_Reference max_match=100 "\((?<Macro_Args>.*?)\)"
| makemv delim="," Macro_Args
| eval Macro_Args_Count = mvcount(Macro_Args)
| eval Macro_Title = if (isnull(Macro_Args_Count), Macro_Name, Macro_Name . "(" . Macro_Args_Count . ")")
| eval Macro_Title = if(((Macro_Title == "") OR isnull(Macro_Title)), "no-macro-title", Macro_Title)
| fields - Macro_Reference1 Macro_Name Macro_Args Macro_Args_Count
```

Admin Pilot For Splunk - *https://splunkbase.splunk.com/app/6489*

App Highlights - Admin Pilot for Splunk

Macro: *get_lookup_reference(1)*

```
rex field=$field$ max_match=100 "\|\s*inputlookup\s+(?<Input_Lookup>[^|]+)"
| rex field=$field$ max_match=100 "\|\s*from\s+inputlookup:(?<From_Input_Lookup>[^|]+)"
| rex field=$field$ max_match=100 "\|\s*from\s+lookup:(?<From_Lookup>[^|]+)"
| rex field=$field$ max_match=100 "\|\s*outputlookup\s+(?<Output_Lookup>[^|]+)"
| rex field=$field$ max_match=100 "\|\s*lookup\s+(?<Lookup_Lookup>[^|\s]+)"
| eval Input_Lookup = "Input_Lookup:".Input_Lookup , From_Input_Lookup = "From_Input_Lookup:".From_Input_Lookup, From_Lookup =
"From_Lookup:".From_Lookup, Output_Lookup = "Output_Lookup:".Output_Lookup, Lookup_Lookup = "Lookup_Lookup:".Lookup_Lookup
| eval Lookup_Reference=mvsort(mvdedup(lower(mvappend(Lookup_Lookup,Input_Lookup,From_Lookup,From_Input_Lookup,Output_Lookup))))
| rex field=Lookup_Reference mode=sed
"s/\"|append=\w+|create_empty=\w+|createinapp=\w+|override_if_empty=\w+|event_time_field=\w+|output_format=\w+|local=\w+|update=\w+|key_field=\w+
|enabled=\w+|max=\w+|type=\w+|\s+where\s+.*|\$//g"
```| rex field=Lookup_Reference mode=sed "s/(\s|\]).*$//g" ")```
| eval Lookup_Reference=if(((Lookup_Reference == "") OR isnull(Lookup_Reference)),"no-lookup-reference", mvsort(mvdedup(trim(Lookup_Reference))))
| fields - Input_Lookup,From_Input_Lookup,From_Lookup,Output_Lookup,Lookup_Lookup
| fillnull value="no-lookup-reference" Lookup_Reference
```

**Admin Pilot For Splunk -** *https://splunkbase.splunk.com/app/6489*
```

App Highlights - Admin Pilot for Splunk

Macro: *get_datamodel_reference(1)*

```
rex field=$field$ max_match=100 "[fF][rR][oO][mM]\s*[dD][aA][tT][aA][mM][oO][dD][eE][lL][:=](?<Datamodel_Reference1>.*?)\s"
| rex field=$field$ max_match=100 "\|\s*(datamodel|datamodelsimple)\s+(?<Datamodel_Reference2>.*?)\s"
| eval Datamodel_Reference=coalesce(Datamodel_Reference1,Datamodel_Reference2)
| rex field=Datamodel_Reference mode=sed "s/\"//g"
| eval Datamodel_Reference = mvfilter( ! match(Datamodel_Reference, "^\$|type=|^\|") )
| eval Datamodel_Reference=if(((Datamodel_Reference == "") OR isnull(Datamodel_Reference)),"no-datamodel-reference",
mvdedup(mvsort(Datamodel_Reference)))
| fields - Datamodel_Reference1 Datamodel_Reference2
| fillnull value="no-datamodel-reference" Datamodel_Reference
```

Admin Pilot For Splunk - *https://splunkbase.splunk.com/app/6489*

App Highlights - Admin Pilot for Splunk

Macro: *get_dashboard_reference(1)*

```
rex field=$field$ max_match=100 "href=\"(?<Dashboard_Reference>\w+)\">"
| eval Dashboard_Reference=mvdedup(mvsort(Dashboard_Reference)), Dashboard_Reference=if(((Dashboard_Reference == "")
OR isnull(Dashboard_Reference)),"no-dashboard-reference",mvdedup(mvsort(Dashboard_Reference)))
| fillnull value="no-dashboard-reference" Dashboard_Reference
```

Admin Pilot For Splunk - *https://splunkbase.splunk.com/app/6489*

App Highlights - Alerts for Splunk Admins

Using regex to extract index and sourcetypes from SPL queries.

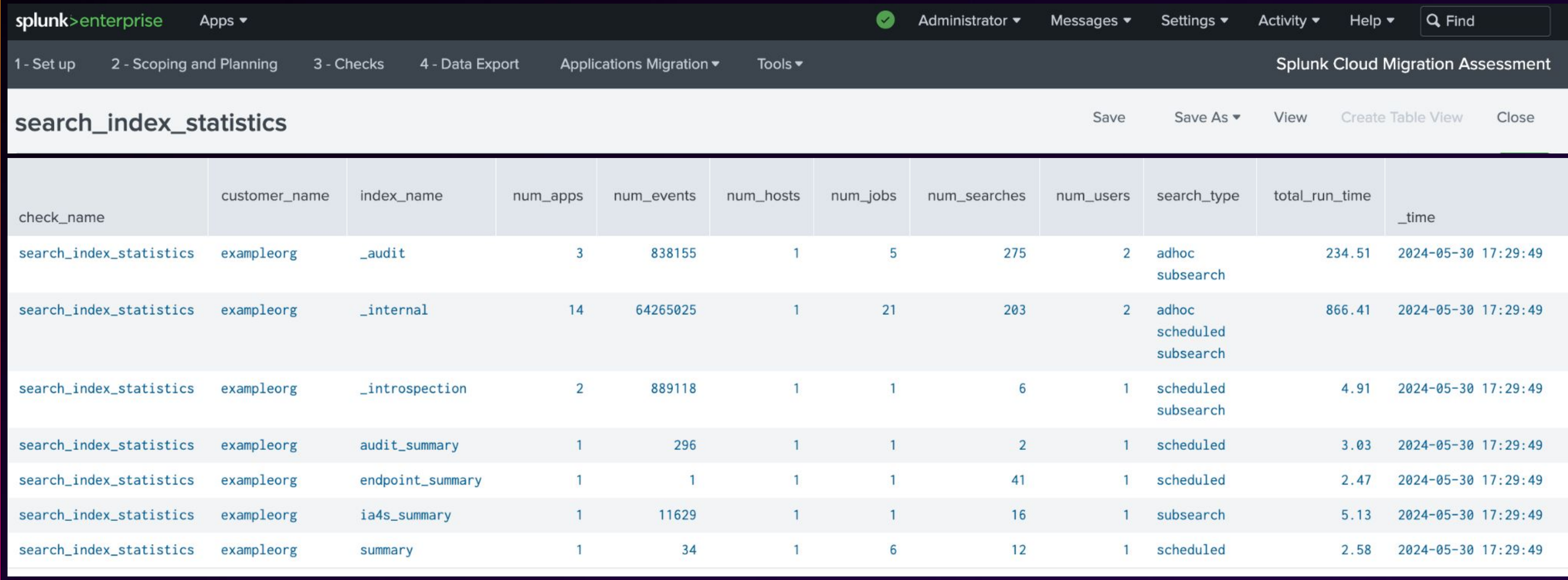
Notable Savedsearches in Alerts for Splunk Admins for data source usage:

- SearchHeadLevel - indexes per savedsearch
- SearchHeadLevel - Search Queries summary exact match
- IndexerLevel - RemoteSearches Indexes Stats
- IndexerLevel - RemoteSearches Indexes Stats Wilcard
- IndexerLevel - RemoteSearches - lookup usage
- IndexerLevel - DataModel Acceleration - Indexes in use
- SearchHeadLevel - Scheduled searches not specifying an index
- SearchHeadLevel - Scheduled searches not specifying an index macro version
- SearchHeadLevel - Users exceeding the disk quota
- SearchHeadLevel - Splunk Users Violating the Search Quota

Alerts for Splunk Admins - https://splunkbase.splunk.com/app/3796

App Highlights - SCMA (Migration Assessment)

Savedsearch: *search_index_statistics*



check_name	customer_name	index_name	num_apps	num_events	num_hosts	num_jobs	num_searches	num_users	search_type	total_run_time	_time
search_index_statistics	exampleorg	_audit	3	838155	1	5	275	2	adhoc subsearch	234.51	2024-05-30 17:29:49
search_index_statistics	exampleorg	_internal	14	64265025	1	21	203	2	adhoc scheduled subsearch	866.41	2024-05-30 17:29:49
search_index_statistics	exampleorg	_introspection	2	889118	1	1	6	1	scheduled subsearch	4.91	2024-05-30 17:29:49
search_index_statistics	exampleorg	audit_summary	1	296	1	1	2	1	scheduled	3.03	2024-05-30 17:29:49
search_index_statistics	exampleorg	endpoint_summary	1	1	1	1	41	1	scheduled	2.47	2024-05-30 17:29:49
search_index_statistics	exampleorg	ia4s_summary	1	11629	1	1	16	1	subsearch	5.13	2024-05-30 17:29:49
search_index_statistics	exampleorg	summary	1	34	1	6	12	1	scheduled	2.58	2024-05-30 17:29:49

Splunk Cloud Migration Assessment - https://splunkbase.splunk.com/app/4974

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

```
| rest
/servicesNS/-/-/search/jobs
| fields sid, …
```

Using REST:

`/search/jobs`

search/jobs

```
https://<host>:<mPort>/services/search/jobs
```

List search jobs.

For more information about this and other search endpoints, see Creating searches using the REST API in the *REST API Tutorial*.

keywords	All positive keywords used by this search. A positive keyword is a keyword that is not in a NOT clause.

Warning: Lots of fields!

- 1,563 fields returned in our SplCloud env

Use `| fields` or `| table`

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

```
| rest
/servicesNS/-/-/search/jobs
| fields sid, …
```

Using REST:
/search/jobs

search/jobs

```
https://<host>:<mPort>/services/search/jobs
```

List search jobs.

For more information about this and other search endpoints, see Creating searches using the REST API in the *REST API Tutorial*.

keywords	All positive keywords used by this search. A positive keyword is a keyword that is not in a NOT clause.

eventSearch	keywords
search (sourcetype="scheduler" (NOT index="foo" OR NOT sourcetype=bar) (index="_internal" OR index="aws" OR index="botsv4"))	index::_internal index::aws index::botsv4 sourcetype::scheduler

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

```
| rest
/servicesNS/-/-/search/jobs
| fields sid, …
```

Using REST:

`/search/jobs`

Benefits:

- Very useful information not found in default indexes
- *keywords* field contains exactly the info we're seeking

Challenges:

- Only explicit terms - index=foo* returns index::foo*
- Only available while the job artifact is alive
  - Write output to summary to get insight over time
- Subsearch artifacts expire very quickly, do not receive parent search TTL settings
- Takes time to build a baseline of your environment

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

/search/jobs

SIDs, Subsearches, *eventSearch*, & *keywords*

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
   search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| search NOT title="*/search/jobs*"
| rename title AS searchQuery
| table label, sid, searchQuery, eventSearch, keywords
```

label	sid	searchQuery	eventSearch	keywords
Sample Data Generation - Add NOT	scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717122660_34555	search index=_internal OR index=_telemetry NOT sourcetype=splunkd_ui_access NOT sourcetype="*modular_input*" NOT index=_introspection | stats count by index | append [search index=_audit sourcetype=audittrail | stats count by index]	search ((index=_internal OR index=_telemetry) NOT sourcetype=splunkd_ui_access NOT sourcetype="*modular_input*" NOT index=_introspection)	index::_internal index::_telemetry
	subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717122660_34555_1717122702.19	| search (index=_audit sourcetype=audittrail) | stats count by index	search (index=_audit sourcetype=audittrail)	index::_audit sourcetype::audittrail

/search/jobs

SIDs, Subsearches, *eventSearch*, & *keywords*

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
   search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| search NOT title="*/search/jobs*"
| rename title AS searchQuery
| table label, sid, searchQuery, eventSearch, keywords
```

label	sid	searchQuery	eventSearch	keywords
Sample Data Generation - Add NOT	scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717122660_34555	search index=_internal OR index=_telemetry NOT sourcetype=splunkd_ui_access NOT sourcetype="*modular_input*" NOT index=_introspection | stats count by index | append [search index=_audit sourcetype=audittrail | stats count by index]	search ((index=_internal OR index=_telemetry) NOT sourcetype=splunkd_ui_access NOT sourcetype="*modular_input*" NOT index=_introspection)	index::_internal index::_telemetry
	subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717122660_34555_1717122702.19	| search (index=_audit sourcetype=audittrail) | stats count by index	search (index=_audit sourcetype=audittrail)	index::_audit sourcetype::audittrail

/search/jobs

SIDs, Subsearches, *eventSearch*, & *keywords*

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
   search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| search NOT title="*/search/jobs*"
| rename title AS searchQuery
| table label, sid, searchQuery, eventSearch, keywords
```

label	sid	searchQuery	eventSearch	keywords
Sample Data Generation - Add NOT	scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717122660_34555	search index=_internal OR index=_telemetry NOT sourcetype=splunkd_ui_access NOT sourcetype="*modular_input*" NOT index=_introspection | stats count by index | append [search index=_audit sourcetype=audittrail | stats count by index]	search ((index=_internal OR index=_telemetry) NOT sourcetype=splunkd_ui_access NOT sourcetype="*modular_input*" NOT index=_introspection)	index::_internal index::_telemetry
	subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717122660_34555_1717122702.19	| search (index=_audit sourcetype=audittrail) | stats count by index	search (index=_audit sourcetype=audittrail)	index::_audit sourcetype::audittrail

/search/jobs

SIDs, Subsearches, *eventSearch*, & *keywords*

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
   search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| search NOT title="*/search/jobs*" | rename title AS searchQuery
``` Handle all empty-string fields returned from API by explicitly setting to null if length is less than 1 ```
| foreach * [| eval <<FIELD>> = if( isnull('<<FIELD>>') OR len('<<FIELD>>') < 1, null(), '<<FIELD>>')]
| `get_normalized_search_id(sid)`
| eventstats values(label) AS label, values(keywords) AS all_keywords BY search_id_normalized
| table label, search_id_normalized, sid, searchQuery, eventSearch, all_keywords
```

label	search_id_normalized	sid	searchQuery	keywords	all_keywords
Sample Data Generation - Add NOT	1717122660_34555	scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717122660_34555	search index=_internal OR index=_telemetry NOT sourcetype=splunkd_ui_access NOT [...] | stats count by index | append [search index=_audit sourcetype=audittrail | stats count by index]	index::_internal index::_telemetry	index::_audit sourcetype::audittrail index::_internal index::_telemetry
Sample Data Generation - Add NOT	1717122660_34555	subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717122660_34555_1717122702.19	| search (index=_audit sourcetype=audittrail) | stats count by index	index::_audit sourcetype::audittrail	index::_audit sourcetype::audittrail index::_internal index::_telemetry
```

/search/jobs

Regex on SPL - Data Source Usage from /search/jobs *keywords*

```
[...]
| `get_normalized_search_id(sid)`
| eventstats min(eval(len(sid))) AS min_sid_length, values(label) AS label, values(keywords) AS all_keywords
            BY search_id_normalized
| eval isPrimarySID = if( len(sid)=min_sid_length, 1, 0 ) | fields - min_sid_length
| rex field=all_keywords max_match=0 "index::(?<index_keywords>[^\s\n]+)"
| rex field=all_keywords max_match=0  "sourcetype::(?<sourcetype_keywords>[^\s\n]+)"
| table label, search_id_normalized, searchQuery, isPrimarySID, all_keywords, index_keywords, sourcetype_keywords
```

/search/jobs

Regex on SPL - Data Source Usage from /search/jobs *keywords*

```
[...]
| `get_normalized_search_id(sid)`
| eventstats min(eval(len(sid))) AS min_sid_length, values(label) AS label, values(keywords) AS all_keywords
            BY search_id_normalized
| eval isPrimarySID = if( len(sid)=min_sid_length, 1, 0 ) | fields - min_sid_length
| rex field=all_keywords max_match=0 "index::(?<index_keywords>[^\s\n]+)"
| rex field=all_keywords max_match=0  "sourcetype::(?<sourcetype_keywords>[^\s\n]+)"
| table label, search_id_normalized, searchQuery, isPrimarySID, all_keywords, index_keywords, sourcetype_keywords
```

label	search_id_normalized	searchQuery	isPrimarySID	all_keywords	index_keywords	sourcetype_keywords
Sample Data Generation - Add NOT	1717185840_36053	search index=_internal OR index=_telemet* NOT sourcetype=splunkd_ui_access NOT index=_introspection | stats count by index | append [search index=_audit sourcetype=audittrail | stats count by index]	1	index::_audit sourcetype::audittrail index::_internal index::_telemet*	_audit _internal _telemet*	audittrail
Sample Data Generation - Add NOT	1717185840_36053	| search (index=_audit sourcetype=audittrail) | stats count by index	0	index::_audit sourcetype::audittrail index::_internal index::_telemet*	_audit _internal _telemet*	audittrail

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

```
| rest
/servicesNS/-/-/search/jobs
| fields sid, ...
```

Using REST: /search/jobs

Writing /search/jobs to summary index:

PopulateSummary - Search Jobs REST Data Collection - No search.log

Save | Save As | View

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
    search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| rename title AS searchQuery, sid AS search_id, label AS savedsearch_label, eai:acl.app AS appName, eai:acl.owner AS owner, eventFieldCount AS rawEventFieldsReturned
    ``` FILTER - Drop any search jobs you know you don't want - Defaults are any search/jobs query and either of the presentation collection queries ```
| search NOT searchQuery IN ("*/search/jobs*", "*search_log_events*", "*search_jobs_rest_collection*", "*search_jobs_search_log_data*")
    ``` FILTER - Drop any SIDs that have already been written to the summary location
    Subsearch passes compiled filter string: splunk_server=<host> AND search_id IN (...)```
| search NOT [search index=summary sourcetype=search_jobs_rest_collection earliest=-4h@h latest=+4h@h
    | rex field=_raw ", search_id=\"?(?<search_id>[^\",]+)\"?,"
    | rex field=_raw "orig_splunk_server=\"(?<orig_splunk_server>[^\"]+)"
    | stats values(search_id) AS search_id_list BY orig_splunk_server
    | eval "Combined Subsearch Filter for REST API Collection Job" = "( splunk_server=\"" . orig_splunk_server . "\" AND search_id IN (\"" . mvjoin(search_id_list, "\", \"") . "\") )"
    | rename "Combined Subsearch Filter for REST API Collection Job" AS search
    | table search]
    ``` FILTER - to search artifacts published within last 4h - Timerange here should be same as in the search NOT filter above/below. ```
| eval _time = strptime(published, "%Y-%m-%dT%H:%M:%S.%3N")
| where _time >= relative_time(now(), "-4h")
    ``` Output to summary index will not include original SPL, if you want the original SPL join on sourcetype=audittrail with search_id ```
| table _time, splunk_server, search_id, savedsearch_label, appName, owner, dispatchState, keywords, runDuration, searchEarliestTime, searchLatestTime, diskUsage, rawEventFieldsReturned, resultCount,
    eventCount, scanCount, dropCount, searchTotalEliminatedBucketsCount, searchTotalBucketsCount, performance.command.search.index.invocations, performance.command.search.kv.duration_secs, priority,
    provenance, dispatchAs, request.ui_dispatch_app, eventSearch, published
    ``` Handle all empty string false-null fields returned from API by explicitly setting to null if length is less than 1 ```
| foreach *
    [| eval <<FIELD>> = if( isnull('<<FIELD>>') OR len('<<FIELD>>') < 1, null(), '<<FIELD>>')]
    ``` Generate normalized SID to populate savedsearch_label value properly across search artifacts ```
| `get_normalized_search_id(search_id)`
    ``` Aggregate key fields across SIDs using normalized SID - Note: min_sid_length is used to identify primary search ```
| eventstats min(eval(len(search_id))) AS min_sid_length, values(savedsearch_label) AS savedsearch_label, values(keywords) AS all_keywords BY search_id_normalized
| eval isPrimarySID = if( len(search_id)=min_sid_length AND !match(search_id, "^(remote|subsearch)"), 1, 0 )
| fields - min_sid_length
    ``` FILTER - Do not write summary index event if the search is not DONE - wait until this point to ensure label is passed to all subsearch artifacts ```
| search dispatchState="DONE"
    ``` Clean/Modify data as needed in preparation to write to summary index ```
| eval endpoint = "/servicesNS/-/-/search/jobs"
| foreach searchEarliestTime, searchLatestTime, runDuration
    [| eval <<FIELD>> = round('<<FIELD>>', 0)]
| addinfo | fields - info_*time | rename info_sid AS population_sid
    ``` Write output from search/jobs base collection to Summary Index - use appendpipe to avoid writing summary events in ad-hoc search ```
| appendpipe
    [| where match(population_sid, "^(scheduler|_scheduler_)")
    | collect testmode=f addinfo=f index=summary sourcetype=search_jobs_rest_collection marker="search_jobs_rest_collection_job"
    | where false()]
```

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

```
| rest
/servicesNS/-/-/search/jobs
| fields sid, …
```

Using REST: /search/jobs

Writing /search/jobs to summary index:

Event

05/02/2024 13:29:18 -0400, search_name="PopulateSummary - Search Jobs REST Data Collection - No search.log", search_now=1717349340.000, owner="ryan.wood@exampleorg.com", appName=search, endpoint="/servicesNS/-/-/search/jobs", keywords="index::_audit sourcetype::audittrail", priority=5, diskUsage=147456, dropCount=0, published="2024-05-02T13:29:18.000-04:00", scanCount=2734719, search_id="subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717349280_40383_1717349358.1", searchTotalBucketsCount=36, searchLatestTime=1717348980, searchTotalEliminatedBucketsCount=0, orig_splunk_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com", performance.command.search.index.invocations=36, performance.command.search.kv.duration_secs="0.467", eventCount=2734719, search_id_normalized=1717349280_40383, searchEarliestTime=1716696000, rawEventFieldsReturned=0, eventSearch="search (index=_audit sourcetype=audittrail)", resultCount=1, runDuration=2, savedsearch_label="Sample Data Generation - Add NOT", all_keywords="index::_audit sourcetype::audittrail index::_internal index::_telemet*", isPrimarySID=0, population_sid=scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5f087672533bb90ff_at_1717349340_40384, dispatchState=DONE, search_jobs_rest_collection_job

host = sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com | source = PopulateSummary - Search Jobs REST Data Collection - No search.log | sourcetype = search_jobs_rest_collection

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - auditttrail
  - remote_searches.log
  - /search/jobs REST API

```
| rest
/servicesNS/-/-/search/jobs
| fields sid, …
```

Using REST: /search/jobs

Writing /search/jobs to summary index:

Event

05/02/2024 13:29:18 -0400, search_name="PopulateSummary - Search Jobs REST Data Collection - No search.log", search_now=1717349340.000, owner="ryan.wood@exampleorg.com", appName=search, endpoint="/servicesNS/-/-/search/jobs", keywords="index::_audit sourcetype::audittrail", priority=5, diskUsage=147456, dropCount=0, published="2024-05-02T13:29:18.000-04:00", scanCount=2734719, search_id="subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717349280_40383_1717349358.1", searchTotalBucketsCount=36, searchLatestTime=1717348980, searchTotalEliminatedBucketsCount=0, orig_splunk_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com", performance.command.search.index.invocations=36, performance.command.search.kv.duration_secs="0.467", eventCount=2734719, search_id_normalized=1717349280_40383, searchEarliestTime=1716696000, rawEventFieldsReturned=0, eventSearch="search (index=_audit sourcetype=audittrail)", resultCount=1, runDuration=2, savedsearch_label="Sample Data Generation - Add NOT", all_keywords="index::_audit sourcetype::audittrail index::_internal index::_telemet*", isPrimarySID=0, population_sid=scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5f087672533bb90ff_at_1717349340_40384, dispatchState=DONE, search_jobs_rest_collection_job

host = sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com | source = PopulateSummary - Search Jobs REST Data Collection - No search.log | sourcetype = search_jobs_rest_collection

- Population SPL in PDF
  - Change *index=summary* to desired output location
  - Uses techniques shown so far
- Scheduled search running every 2/3 minutes
  - Filters to dispatch times within last 4h
  - Writes 1 "DONE" event per SID, filters SIDs already written

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

/search/jobs

Writing search/jobs to Summary - SPL Functionality Highlights

- Change collect command target index as desired
- See docs on collect for other options

  https://docs.splunk.com/Documentation/Splunk/latest/SearchReference/Collect

```
[...]
| collect testmode=f addinfo=f index=summary sourcetype=search_jobs_rest_collection marker="search_jobs_rest_collection_job"
```

© 2024 SPLUNK INC.

/search/jobs

Writing search/jobs to Summary - SPL Functionality Highlights

- Drop any searches using /search/jobs, as well as any collection searches from the slides.
- Collecting metadata on the jobs collecting metadata should be done with caution and care.
  - Recursion risk

```
[...]
    ``` FILTER - Drop any search jobs you know you don't want - Defaults are any search/jobs query and either of the presentation collection queries ```
| search NOT searchQuery IN ("*/search/jobs*", "*search_log_events*", "*search_jobs_rest_collection*", "*search_jobs_search_log_data*")
```
```

/search/jobs

Writing search/jobs to Summary - SPL Functionality Highlights

- `| search NOT [search` returns a filter string of the splunk_server + search IDs already written
- Timerange for filter subsearch and `| where _time >=` lines should always be the same

Combined Subsearch Filter for REST API Collection Job

```
( splunk_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com" AND search_id IN ("1717344439.160427", "1717344619.160435",
"1717344799.160438", "1717344979.160443", "1717345159.160451", "1717345339.160469", "1717345519.160479", "1717345699.160489",
"1717345879.160502", "1717346059.160512", "scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717344600_40255",
"scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717344900_40261",
"scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717345200_40270",
"scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717345500_40282",
```

/search/jobs

Writing search/jobs to Summary - SPL Functionality Highlights

- `| search NOT [search` returns a filter string of the splunk_server + search IDs already written
- Timerange for subsearch and `| where _time >=` lines should always be the same
- `splunk_server` is auto-renamed to `orig_splunk_server` when collect writes summary events

```
[...]
   ``` FILTER - Drop any SIDs that have already been written to the summary location
   Subsearch passes compiled filter string: splunk_server=<host> AND search_id IN (...)```
| search NOT [search index=summary sourcetype=search_jobs_rest_collection earliest=-4h@h latest=+4h@h
   | rex field=_raw ", search_id=\"?(?<search_id>[^\",]+)\"?,"
   | rex field=_raw "orig_splunk_server=\"(?<orig_splunk_server>[^\"]+)"
   | stats values(search_id) AS search_id_list BY orig_splunk_server
   | eval "Combined Subsearch Filter for REST API Collection Job" =
            "( splunk_server=\"" . orig_splunk_server . "\" AND search_id IN (\"" . mvjoin(search_id_list, "\", \"") . "\") )"
   | rename "Combined Subsearch Filter for REST API Collection Job" AS search
   | table search]
   ``` FILTER - to search artifacts published within last 4h - Timerange here should be same as in the search NOT filter above/below. ```
| eval _time = strptime(published, "%Y-%m-%dT%H:%M:%S.%3N")
| where _time >= relative_time(now(), "-4h")
```

Combined Subsearch Filter for REST API Collection Job
(splunk_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com" AND search_id IN ("1717344439.160427", "1717344619.160435", "1717344799.160438", "1717344979.160443", "1717345159.160451", "1717345339.160469", "1717345519.160479", "1717345699.160489", "1717345879.160502", "1717346059.160512", "scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717344600_40255", "scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717344900_40261", "scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717345200_40270", "scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717345500_40282",

/search/jobs

Writing search/jobs to Summary - SPL Functionality Highlights

- addinfo passes the population search ID to the results
- appendpipe uses that population search ID to avoid writing summary events unless the collection search is a scheduled search - no adhoc accidental writes

```
[...]
| addinfo | fields - info_*time | rename info_sid AS population_sid
   ``` Write output from search/jobs base collection to Summary Index - use appendpipe to avoid ad-hoc writes to summary. ```
| appendpipe
   [| where match(population_sid, "^(scheduler|_scheduler_)")
   | collect testmode=f addinfo=f index=summary sourcetype=search_jobs_rest_collection marker="search_jobs_rest_collection_job"
   | where false()]
```

- If you want to manually write events, remove everything but the | collect

```
[...]
| addinfo | fields - info_*time | rename info_sid AS population_sid
   ``` Write output from search/jobs base collection to Summary Index - use appendpipe to avoid ad-hoc writes to summary. ```
| collect testmode=f addinfo=f index=summary sourcetype=search_jobs_rest_collection marker="search_jobs_rest_collection_job"
```

Data Source Usage

Regex on SPL

- Using regex to extract index and sourcetype from SPL queries in:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

```
| rest
/servicesNS/-/-/search/jobs
| fields sid, ...
```

Using REST: /search/jobs

Tips when collecting /search/jobs data:

- Run it frequently, make it efficient.
  - Only shows existing artifacts at runtime - ephemeral data
  - Filter to the fields you want:
    - Reference /search/jobs documentation
    - PDF contains example with notable fields
    - Handle empty strings - foreach pattern
- Recursion Risk - filter out any searches using /search/jobs:

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
| search NOT title="*/search/jobs*"
```

- Only Primary SIDs or All SIDs? Either:
  - Merge values across parent/child SIDs, drop child SIDs
  - Write all SIDs to summary, merge later when searching
- Be mindful of `dispatchState`
  - Subsearches can be "DONE" while primary is "RUNNING"

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

Utility SPL - search/jobs REST Base Collection

SPL for returning a subset of interesting fields and information for search job artifacts.

```
| rest /servicesNS/-/-/search/jobs splunk_server=* search="sid!=SummaryDirector*" search="sid!=rt_*" search="sid!=rsa_*"
| search NOT title="*/search/jobs*"
| rename title AS searchQuery, eai:acl.app AS appName, eai:acl.owner AS owner, eventFieldCount AS rawEventFieldsReturned
| fields sid, label, appName, owner, dispatchState, keywords, runDuration, searchEarliestTime, searchLatestTime, earliestTime, latestTime,
diskUsage, rawEventFieldsReturned, resultCount, eventCount, scanCount, dropCount, searchTotalEliminatedBucketsCount,
searchTotalBucketsCount, performance.command.search.index.invocations, performance.command.search.kv.duration_secs, priority, provenance,
dispatchAs, request.ui_dispatch_app, published, eventSearch, searchQuery
| eval _time = strptime(published, "%Y-%m-%dT%H:%M:%S.%3N")
   ``` Handle all empty string false-null fields returned from API by explicitly setting to null if length is less than 1 ```
| foreach *
    [| eval <<FIELD>> = if( isnull('<<FIELD>>') OR len('<<FIELD>>') < 1, null(), '<<FIELD>>')]
``` Generate normalized SID to populate label value properly across search artifacts -
   For macro definition see slide: Macro: get_normalized_search_id(1)```
| `get_normalized_search_id(sid)`
``` Aggregate desired fields across Primary/Secondary SIDs using normalized SID
   Note: min_sid_length is used to identify primary search ```
| eventstats min(eval(len(sid))) AS min_sid_length, values(label) AS label, values(keywords) AS all_keywords BY search_id_normalized
| eval isPrimarySID = if( len(sid)=min_sid_length AND !match(sid, "^(remote|subsearch)"), 1, 0 ) | fields - min_sid_length
| table _time, label, sid, search_id_normalized, isPrimarySID, appName, owner, dispatchState, keywords, *_keywords, *
| sort 0 search_id_normalized, -isPrimarySID
```

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs
```

Utility SPL - /search/jobs Regex Identification

Identifying index & sourcetype usage via /search/jobs *keywords* field

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
   search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| search NOT title="*/search/jobs*"
| fields title, published, label, sid, eai:acl.app, eai:acl.owner, dispatchState, keywords
| rename title AS searchQuery, eai:acl.app AS appName, eai:acl.owner AS owner
| eval _time = strptime(published, "%Y-%m-%dT%H:%M:%S.%3N")
``` Handle all empty string false-null fields returned from API by explicitly setting to null if length is less than 1 ```
| foreach * [| eval <<FIELD>> = if( isnull('<<FIELD>>') OR len('<<FIELD>>') < 1, null(), '<<FIELD>>')]
``` Generate normalized SID to populate label value properly across search artifacts ```
| `get_normalized_search_id(sid)`
``` Aggregate desired fields across Primary/Secondary SIDs using normalized SID
   Note: min_sid_length is used to identify primary search ```
| eventstats min(eval(len(sid))) AS min_sid_length, values(label) AS label, values(keywords) AS all_keywords BY search_id_normalized
| eval isPrimarySID = if( len(sid)=min_sid_length AND !match(sid, "^(remote|subsearch)"), 1, 0 )
| fields - min_sid_length
``` Extract keywords references ```
| rex field=all_keywords max_match=0 "index::(?<index_keywords>[^\s\n]+)"
| rex field=all_keywords max_match=0  "sourcetype::(?<sourcetype_keywords>[^\s\n]+)"
| rex field=all_keywords max_match=0  "source::(?<source_keywords>[^\s\n]+)"
| table _time, label, search_id_normalized, isPrimarySID, appName, owner, *_keywords, sid, searchQuery
```

https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

Utility SPL - /search/jobs Summary Population

SPL for collecting /search/jobs information to write to summary index. Schedule frequently, every ~2/3min ideally

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
   search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| rename title AS searchQuery, sid AS search_id, label AS savedsearch_label, eai:acl.app AS appName, eai:acl.owner AS owner, eventFieldCount AS rawEventFieldsReturned
   ``` FILTER - Drop any search jobs you know you don't want - Defaults are any search/jobs query and either of the presentation collection queries ```
| search NOT searchQuery IN ("*/search/jobs*", "*search_log_events*", "*search_jobs_rest_collection*", "*search_jobs_search_log_data*")
   ``` FILTER - Drop any SIDs that have already been written to the summary location
   Subsearch passes compiled filter string: splunk_server=<host> AND search_id IN (...)```
| search NOT [search index=summary sourcetype=search_jobs_rest_collection earliest=-4h@h latest=+4h@h
   | rex field=_raw ", search_id=\"?(?<search_id>[^\",]+)\"?,"
   | rex field=_raw "orig_splunk_server=\"(?<orig_splunk_server>[^\"]+)"
   | stats values(search_id) AS search_id_list BY orig_splunk_server
   | eval "Combined Subsearch Filter for REST API Collection Job" = "( splunk_server=\"" . orig_splunk_server . "\" AND search_id IN (\"" . mvjoin(search_id_list, "\", \"") . "\") )"
   | rename "Combined Subsearch Filter for REST API Collection Job" AS search
   | table search]
   ``` FILTER - to search artifacts published within last 4h - Timerange here should be same as in the search NOT filter above/below. ```
| eval _time = strptime(published, "%Y-%m-%dT%H:%M:%S.%3N")
| where _time >= relative_time(now(), "-4h")
   ``` Output to summary index will not include original SPL, if you want the original SPL join on sourcetype=audittrail with search_id ```
| table _time, splunk_server, search_id, savedsearch_label, appName, owner, dispatchState, keywords, runDuration, searchEarliestTime, searchLatestTime, diskUsage, rawEventFieldsReturned, resultCount, eventCount, scanCount, dropCount,
searchTotalEliminatedBucketsCount, searchTotalBucketsCount, performance.command.search.index.invocations, performance.command.search.kv.duration_secs, priority, provenance, dispatchAs, request.ui_dispatch_app, eventSearch, published
   ``` Handle all empty string false-null fields returned from API by explicitly setting to null if length is less than 1 ```
| foreach *
   [| eval <<FIELD>> = if( isnull('<<FIELD>>') OR len('<<FIELD>>') < 1, null(), '<<FIELD>>')]
   ``` Generate normalized SID to populate savedsearch_label value properly across search artifacts ```
| `get_normalized_search_id(search_id)`
   ``` Aggregate key fields across SIDs using normalized SID - Note: min_sid_length is used to identify primary search ```
| eventstats min(eval(len(search_id))) AS min_sid_length, values(savedsearch_label) AS savedsearch_label, values(keywords) AS all_keywords BY search_id_normalized
| eval isPrimarySID = if( len(search_id)=min_sid_length AND !match(search_id, "^(remote|subsearch)"), 1, 0 )
| fields - min_sid_length
   ``` FILTER - Do not write summary index event if the search is not DONE - wait until this point to ensure label is passed to all subsearch artifacts ```
| search dispatchState="DONE"
   ``` Clean/Modify data as needed in preparation to write to summary index ```
| eval endpoint = "/servicesNS/-/-/search/jobs"
| foreach searchEarliestTime, searchLatestTime, runDuration
   [| eval <<FIELD>> = round('<<FIELD>>', 0)]
| addinfo | fields - info_*time | rename info_sid AS population_sid
   ``` Write output from search/jobs base collection to Summary Index - use appendpipe to avoid writing summary events in ad-hoc search ```
| appendpipe
   [| where match(population_sid, "^(scheduler|_scheduler_)")
   | collect testmode=f addinfo=f index=summary sourcetype=search_jobs_rest_collection marker="search_jobs_rest_collection_job"
   | where false()]
```


https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs

Data Source Usage

Using Summary

- search/jobs *keywords*

  OR

- search.log *index* messages

Using Summary Population for Index Usage

```
index=summary sourcetype=search_jobs_rest_collection isPrimarySID=1
| rex field=_raw ", search_id=\"?(?<search_id>[^\",]+)\"?,"
| rex field=_raw ", savedsearch_label=\"?(?<savedsearch_label>[^\",]+)\"?,"
| rex field=all_keywords max_match=0 "index::(?<index_keywords>[^\s\n]+)"
| rex field=all_keywords max_match=0  "sourcetype::(?<sourcetype_keywords>[^\s\n]+)"
| rex field=all_keywords max_match=0  "source::(?<source_keywords>[^\s\n]+)"
| stats dc(search_id) AS search_count, values(*_keywords) AS *_keywords
BY savedsearch_label
```

savedsearch_label	search_count	all_keywords	index_keywords	source_keywords	sourcetype_keywords
SIM Splunkd - Data Parsing Issues	4	component::aggregatorminingprocessor component::dateparserverbose component::linebreakingprocessor host::idx*.*splunk*.* host::si*.*splunk*.* index::_internal log_level::error log_level::warn source::*splunkd.log*	_internal	*splunkd.log*	
Sample Data Generation - Add NOT	37	index::_audit sourcetype::audittrail index::_internal index::_telemet* index::_internal index::_telemet*	_audit _internal _telemet*		audittrail
new_user_created	25	action::create_user index::_audit	_audit		

What about search.log?

Bring on
the Verbosity

Usecase - Data Source Usage

Approached three ways:

Instrumentation

- sourcetype=audittrail

sourcetype_count__<sourcetype>

Challenges:

- Imprecise - Not all search types

Regex on SPL

- Using regex to extract index and sourcetype from:
  - audittrail
  - remote_searches.log
  - /search/jobs REST API

Challenges:

- Imprecise due to complexity
- Schema changes over time

search.log

- Ingesting search.log files

  OR
- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

Challenges:

- Imprecise - Point-in-Time
  - Ephemeral Job Artifacts
- Ingest vs Visibility Trade-off

Data Source Usage

search.log

- Ingesting search.log files

  OR

- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

search.log

IndexScopedSearch is called for index = <index>

- Splunk Enterprise
- One index noted per line

Event

```
05-02-2024 23:55:02.623 INFO
IndexScopedSearch [1319756 localCollectorThread] -
IndexScopedSearch is called for index = notable, et = 1714532400.000000000, lt = 1714708500.000000000, index_et
= -9223372036854775808.000000000, index_lt = 9223372036854775807.999999000, noRead = FALSE
```

host = examplehost | source = /opt/splunk/var/run/splunk/dispatch/scheduler__admin__SplunkEnterpriseSecur...

```
05-02-2024 23:54:01.272 INFO
IndexScopedSearch [1315741 localCollectorThread] -
IndexScopedSearch is called for index = wineventlog_security, et = 1714707540.000000000, lt = 1714708440.0000000
00, index_et = 1714707540.000000000, index_lt = 1714708440.000000000, noRead = FALSE
```

host = examplehost | source = /opt/splunk/var/run/splunk/dispatch/scheduler__admin__ThreatHunting__RMD...

Data Source Usage

search.log

- Ingesting search.log files

  OR

- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

search.log

Search requires the following indexes="[<indexes>]"

- Splunk Cloud
- Multiple indexes per line
- Not generated for subsearches

Event

06-02-2024 19:30:00.291 INFO DispatchCommandProcessor [3088817 RunDispatch] - Search requires the following indexes="[_audit,_configtracker,_dsappevent,_dsclient,_dsphonehome,_internal,_introspection,_telemetry,_thefishbucket,cim_modactions,edge_hub_logs,edge_hub_modbus,edge_hub_opcua,edge_hub_snmp,history,main,splunk_app_ar_metrics,summary]" ~~~META:, app="Splunk_SA_CIM", endpoint=".../search.log", search_id="scheduler__nobody_U3BsdW5rX1NBX0NJTQ__RMD554e4d8a17d0ef86d_at_1717356600_40593", log_src_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com", owner="splunk-system-user", savedsearch_label="_ACCELERATE_DM_Splunk_SA_CIM_Web_ACCELERATE_", search_jobs_rest_collection_search_log_data

host = sh-i-000e778792a422e91.examplehost-jg.splunkcloud.com
source = PopulateSummary - Search Jobs REST Data Collection WITH search.log Collecti...
sourcetype = search_jobs_search_log_events

06-02-2024 19:29:17.256 INFO DispatchCommandProcessor [3043911 RunDispatch] - Search requires the following indexes="[_internal,_telemetry]" ~~~META:, app="search", endpoint=".../search.log", search_id="scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717356480_40591", log_src_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com", owner="ryan.wood@exampleorg.com", savedsearch_label="Sample Data Generation - Add NOT", search_jobs_rest_collection_search_log_data

host = sh-i-000e778792a422e91.examplehost-jg.splunkcloud.com
source = PopulateSummary - Search Jobs REST Data Collection WITH search.log Collecti...
sourcetype = search_jobs_search_log_events

Data Source Usage

search.log

- Ingesting search.log files

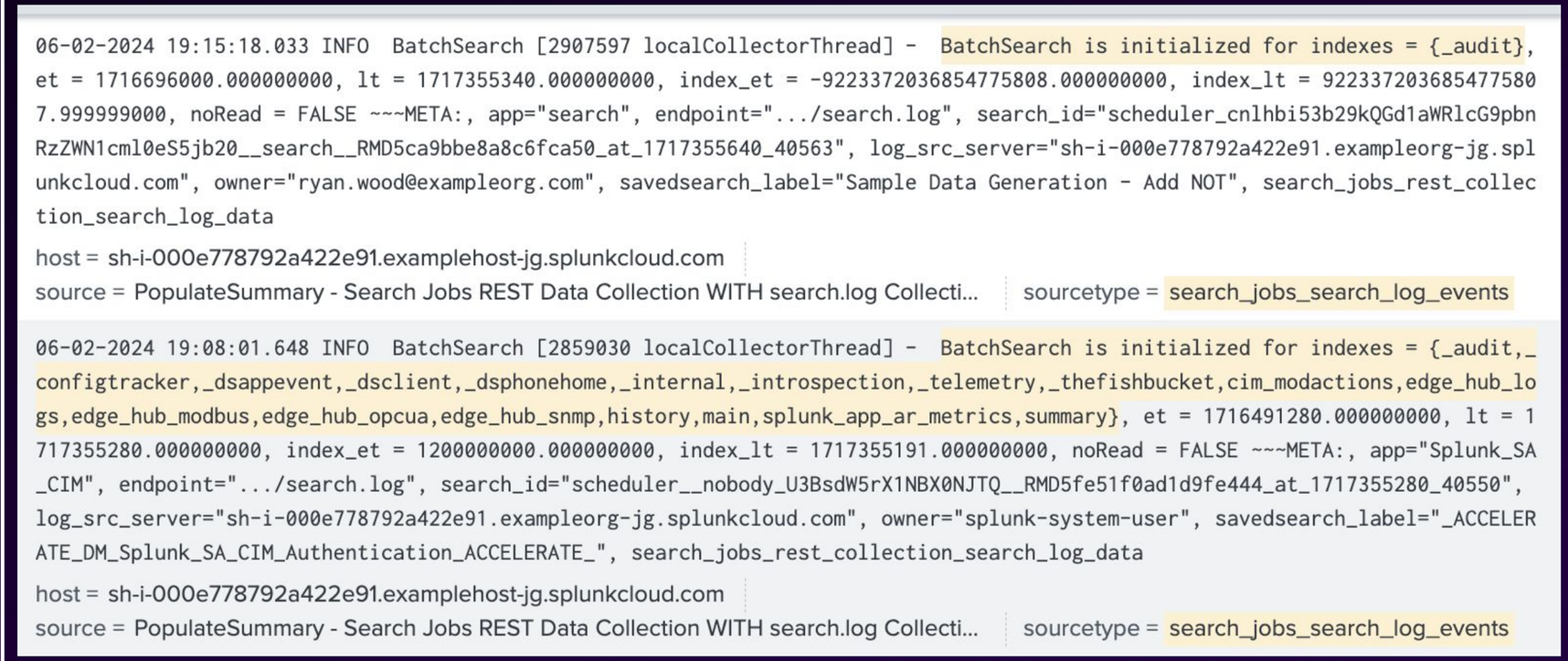
  OR

- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

search.log

BatchSearch is initialized for indexes = {indexes}

- Splunk Enterprise & Splunk Cloud
- Multiple indexes per line
- BatchSearch is a function of search optimization
  - Searches that don't require time-ordered event consideration
    Typically queries with aggregation - stats, timechart, chart, top, etc.



Data Source Usage

search.log - Messages relevant to Data Source Usage

- Our reference environments:
  - Enterprise v9.0.4
  - Enterprise v9.1.2
  - Enterprise v9.2.0
  - Cloud Victoria v9.1.2308

During research, we identified three distinct search.log lines with information we want to use:

`IndexScopedSearch is called for index = <index>`

`Search requires the following indexes="[index]"`

`BatchSearch is initialized for indexes = {index}`

Event

```
05-02-2024 23:54:00.886 INFO  IndexScopedSearch [3292156 localCollectorThread] - IndexScopedSearch is called for index = wineventlog_security, et = 1717040340.000000000, lt = 1717041240.000000000, index_et = 1717040340.000000000, index_lt = 1717041240.000000000, noRead = FALSE
```

Event

```
05-05-2024 03:58:01.045 INFO  DispatchCommandProcessor [1214775 RunDispatch] - Search requires the following indexes="[_audit,_configtracker,_dsappevent,_dsclient,_dsphonehome,_internal,_introspection,_telemetry,_thefishbucket,cim_modactions,edge_hub_logs,edge_hub_modbus,edge_hub_opcua,edge_hub_snmp,history,main,splunk_app_ar_metrics,summary]"
```

Event

```
05-03-2024 03:58:02.394 INFO  BatchSearch [1214890 localCollectorThread] - BatchSearch is initialized for indexes = {_audit,_configtracker,_dsappevent,_dsclient,_dsphonehome,_internal,_introspection,_telemetry,_thefishbucket,cim_modactions,edge_hub_logs,edge_hub_modbus,edge_hub_opcua,edge_hub_snmp,history,main,splunk_app_ar_metrics,summary}, et = 1716177480.000000000, lt = 1717041480.000000000, index_et = 1200000000.000000000, index_lt = 1717041392.000000000, noRead = FALSE
```

Data Source Usage

search.log

- Ingesting search.log files

  OR

- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

Using search.log

Benefits:

- Accurate - works for searches without "index=" in SPL
  - Avoids abstraction problem with default access, eventtypes, macros
  - Works for wildcard references
- Consistent - as of Splunk v9.0 all search artifact logs *should* contain some of the lines in these slides

Challenges:

- Visibility – Similar to `/search/jobs` REST collection
- Ingest size of search.log
- Takes time to build "proper" insight - Point-in-Time Views
  - Users aren't always consistent:
    - Investigation into Incidents/Issues
    - App validation/QA Processes
  - Infrequent-use data, like Compliance

App Highlight: Index Usage Reporting

**Splunk Enterprise Only - Located on Github

Note: this app is not "feature complete" & was built on Splunk 9.0

- Simple app - Ingests search.log files from dispatch directory and writes them to:

```
index=_internal sourcetype=search_logs
```

```
# inputs.conf

# Slurp in all search.log files from the SH this is deployed on
[monitor:///opt/splunk/var/run/splunk/dispatch/*/search.log]
index=_internal
sourcetype=search_logs
```

```
# props.conf

[search_logs]
TIME_PREFIX = ^
MAX_TIMESTAMP_LOOKAHEAD = 25
TIME_FORMAT = %m-%d-%Y %H:%M:%S.%3N
SHOULD_LINEMERGE = false
LINE_BREAKER = ([\r\n]+)
TRUNCATE = 100000
EVENT_BREAKER_ENABLE = TRUE
EVENT_BREAKER = "([\r\n]+)"
```

*Credit to David Paper
https://github.com/dpaper-splunk/public/tree/master/apps/IUR

App Highlight: Index Usage Reporting

**Splunk Enterprise Only - Located on Github

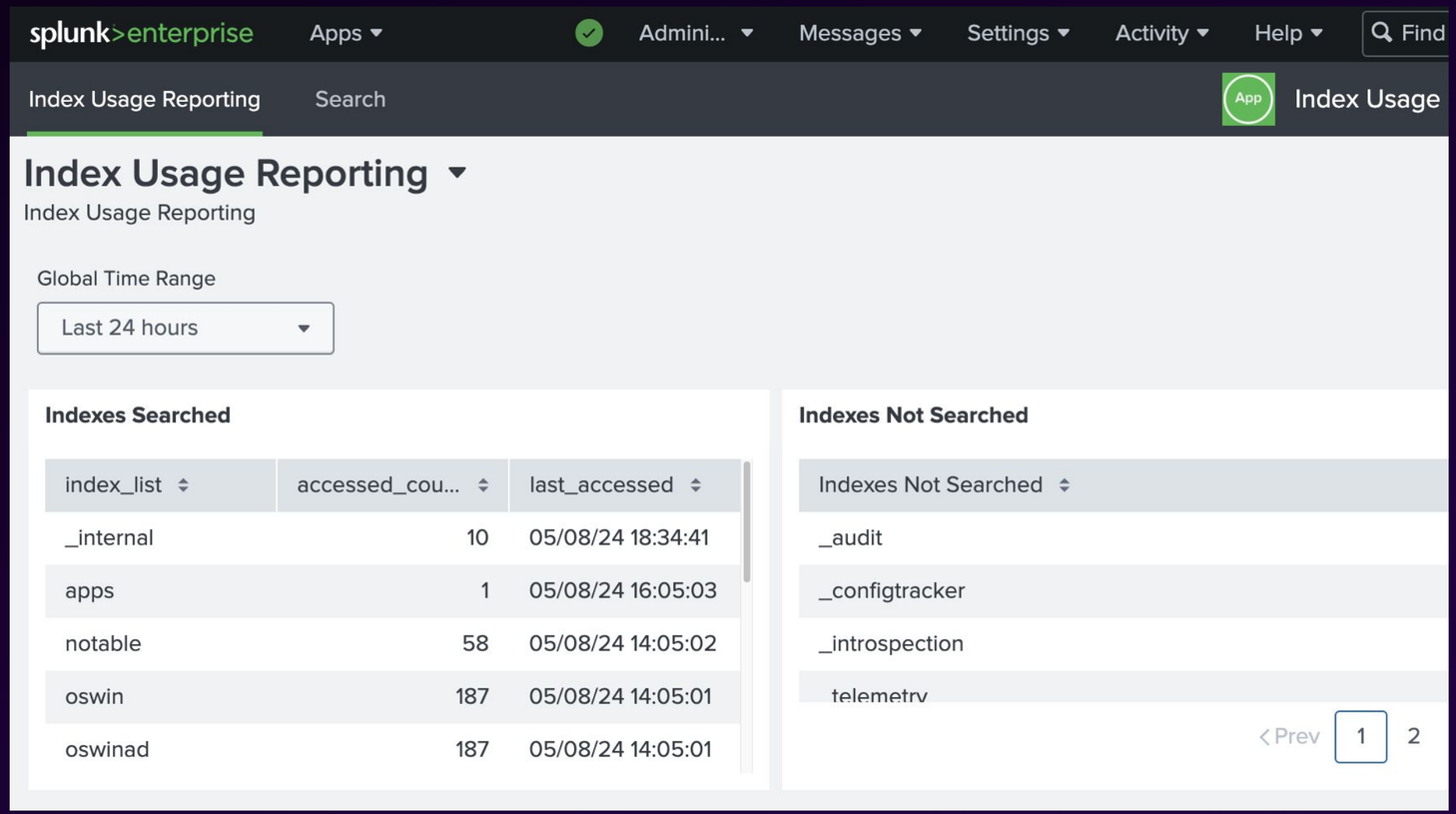
Note: this app is not "feature complete" & was built on Splunk 9.0

- Simple app - Ingests search.log files from dispatch directory and writes them to:

```
index=_internal sourcetype=search_logs
```

- Uses specific search.log line to identify index usage:

"IndexScopedSearch is called"



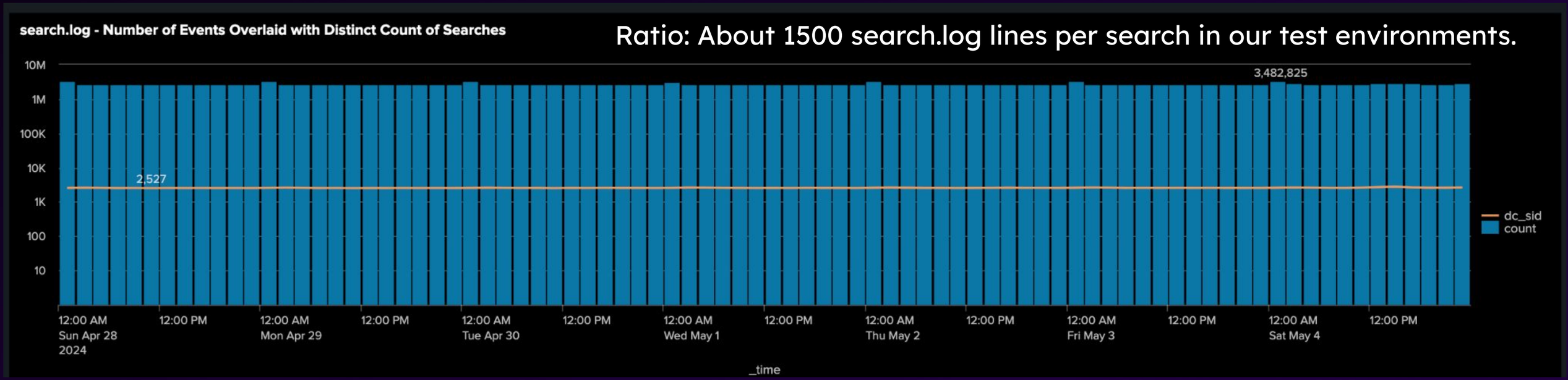
*Credit to David Paper
https://github.com/dpaper-splunk/public/tree/master/apps/IUR

Data Source Usage

Ingesting search.log files

search.log contains a *ton* of useful information about searches

… but it's also a *ton* of data to ingest.



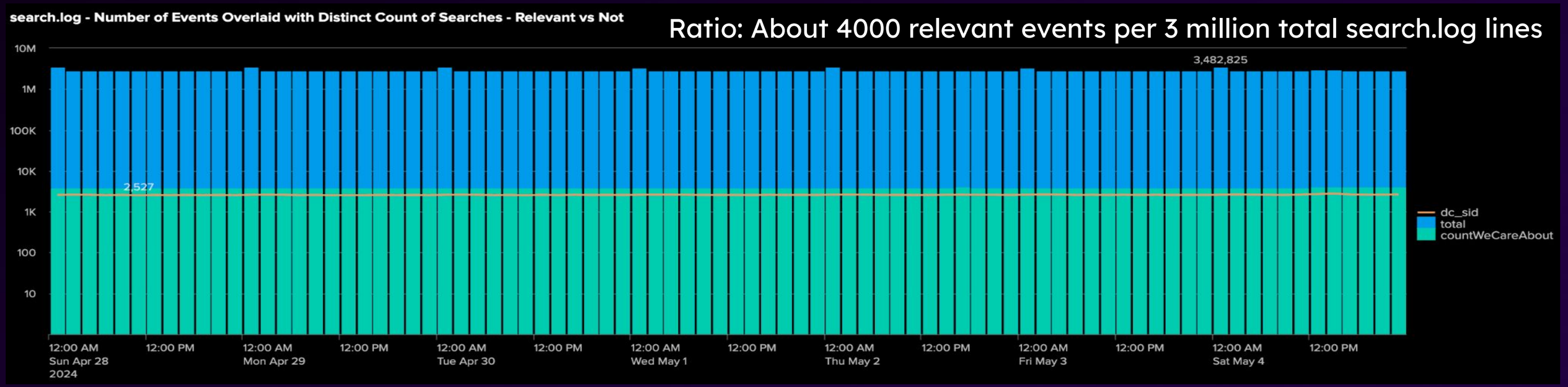
Data Source Usage

Ingesting search.log files

search.log contains a *ton* of useful information about searches

… but it's also a *ton* of data to ingest.

Key Question: Which messages do you care about?



Data Source Usage

search.log

- Ingesting search.log files

  OR

- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

search.log - Using REST

search/jobs/{search_id}/search.log

```
https://<host>:<mPort>/services/search/jobs/{search_id}/search.log
```

Get the `{search_id}` search log.

```
| rest /servicesNS/-/-/search/jobs
   /scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717358160_40636/search.log
   splunk_server=*
| rename value AS search_log_line
| makemv search_log_line tokenizer="([^\n]+)"
| mvexpand search_log_line
| search search_log_line IN ("*Search requires the following*", "*BatchSearch is initialized for indexes*")
```

search_log_line
06-02-2024 19:57:17.228 INFO DispatchCommandProcessor [3145791 RunDispatch] - Search requires the following indexes="[_internal,_telemetry]"
06-02-2024 19:57:17.573 INFO BatchSearch [3145806 localCollectorThread] - BatchSearch is initialized for indexes = {_audit}, et = 1716696000.000000000, lt = 1717357860.000000000, index_et = -9223372036854775808.000000000, index_lt = 9223372036854775807.999999000, noRead = FALSE

Data Source Usage

search.log

- Ingesting search.log files

  OR

- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

search.log - Using REST

Collecting search.log in Cloud via REST

search/jobs/{search_id}/search.log

```
https://<host>:<mPort>/services/search/jobs/{search_id}/search.log
```

Get the `{search_id}` search log.

1. Enumerate the SIDs available using /search/jobs
2. Iterate through each SID to fetch the search.log
3. Filter, then write search.log data to summary index

Data Source Usage

search.log

- Ingesting search.log files

  OR

- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

search.log - Using REST

When writing /search/jobs to summary index:

Event

05/02/2024 13:29:18 -0400, search_name="PopulateSummary - Search Jobs REST Data Collection - No search.log", search_now=1717349340.000, owner="ryan.wood@exampleorg.com", appName=search, endpoint="/servicesNS/-/-/search/jobs", keywords="index::_audit sourcetype::audittrail", priority=5, diskUsage=147456, dropCount=0, published="2024-05-02T13:29:18.000-04:00", scanCount=2734719, search_id="subsearch_scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717349280_40383_1717349358.1", searchTotalBucketsCount=36, searchLatestTime=1717348980, searchTotalEliminatedBucketsCount=0, orig_splunk_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com", performance.command.search.index.invocations=36, performance.command.search.kv.duration_secs="0.467", eventCount=2734719, search_id_normalized=1717349280_40383, searchEarliestTime=1716696000, rawEventFieldsReturned=0, eventSearch="search (index=_audit sourcetype=audittrail)", resultCount=1, runDuration=2, savedsearch_label="Sample Data Generation - Add NOT", all_keywords="index::_audit sourcetype::audittrail index::_internal index::_telemet*", isPrimarySID=0, population_sid=scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5f087672533bb90ff_at_1717349340_40384, dispatchState=DONE, search_jobs_rest_collection_job

host = sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com

source = PopulateSummary - Search Jobs REST Data Collection - No search.log | sourcetype = search_jobs_rest_collection

1. Enumerate the SIDs available using /search/jobs
2. Iterate through each SID to fetch the search.log
3. Filter, then write search.log data to summary index

search.log - Using REST

Collecting search.log in Cloud via REST

Output search.log + meta information to summary index.
Collect base /search/jobs & search.log in one scheduled report!

Event

06-02-2024 20:01:17.748 INFO BatchSearch [3155868 localCollectorThread] - BatchSearch is initialized for indexes = {_audit}, et = 1716696000.000000000, lt = 1717358100.000000000, index_et = -9223372036854775808.000000000, index_lt = 9223372036854775807.999999000, noRead = FALSE ~~~META:, app="search", endpoint=".../search.log", search_id="scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717358400_40644", log_src_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com", owner="ryan.wood@exampleorg.com", savedsearch_label="Sample Data Generation - Add NOT", search_jobs_rest_collection_search_log_data

host = sh-i-000e778792a422e91.examplehost-jg.splunkcloud.com | source = PopulateSummary - Search Jobs REST Data Collection WITH search.log Collecti... | sourcetype = search_jobs_search_log_events

06-02-2024 20:01:17.403 INFO DispatchCommandProcessor [3155853 RunDispatch] - Search requires the following indexes="[_internal,_telemetry]" ~~~META:, app="search", endpoint=".../search.log", search_id="scheduler_cnlhbi53b29kQGd1aWRlcG9pbnRzZWN1cml0eS5jb20__search__RMD5ca9bbe8a8c6fca50_at_1717358400_40644", log_src_server="sh-i-000e778792a422e91.exampleorg-jg.splunkcloud.com", owner="ryan.wood@exampleorg.com", savedsearch_label="Sample Data Generation - Add NOT", search_jobs_rest_collection_search_log_data

host = sh-i-000e778792a422e91.examplehost-jg.splunkcloud.com | source = PopulateSummary - Search Jobs REST Data Collection WITH search.log Collecti... | sourcetype = search_jobs_search_log_events

Data Source Usage

search.log

- Ingesting search.log files

  OR

- Using REST API to:
  - Fetch search.log data from search artifacts
  - Write to summary for analysis

search.log - Using REST

Notes on Collection Utility SPL:

- Default target:

  `index=summary sourcetype=search_jobs_search_log_events`

- Only Primary SIDs - all child SIDs are rolled into Primary search.log
- Recursion Risk - filter out any searches using /search/jobs:

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
| search NOT title="*/search/jobs*"
```

- Filter to what you want, don't write all of search.log to summary
- SPL uses | map - creates new subsearch for every input SID
  - search.log vs search.log.1 requires multiple calls. SPL in PDF covers up to search.log.2
  - Default maxsearches in PDF is 200,000
- Uses foreach to manually construct _raw
  - Original line and Meta fields separated by string "~~~META:"

Data Source Usage

Using Summary

- search/jobs *keywords*

  OR

- search.log *index* messages

Using Summary Population for Index Usage

```
index=summary sourcetype=search_jobs_search_log_events
| rex field=_raw ", search_id=\"?(?<search_id>[^\",]+)\"?,"
| rex field=_raw ", savedsearch_label=\"?(?<savedsearch_label>[^\",]+)\"?,"
| rex field=_raw "^(?<orig_search_log_line>.+?)~~~META:?,\s"
| rex field=orig_search_log_line "IndexScopedSearch is called for index =
(?<index_usage_reference>[^\s,]+)"
| rex field=orig_search_log_line "Search requires the following
indexes\s*=\s*\"\[(?<index_usage_array_reference>[^\]]+)"
| rex field=orig_search_log_line "BatchSearch is initialized for
indexes\s*=\s*\{(?<index_usage_tuple_reference>[^\}]+)"
| eval index_usage_array_reference = split(index_usage_array_reference, ",")
| eval index_usage_tuple_reference = split(index_usage_tuple_reference, ",")
| stats dc(search_id) AS search_count, values(index_usage_*) AS index_usage_* BY savedsearch_label
```

savedsearch_label	search_count	index_usage_array_reference	index_usage_tuple_reference
Sample Data Generation - Add NOT	19	_internal _telemetry	_audit
_ACCELERATE_DM_Splunk_SA_CIM_Authentication_ACCELERATE_	28	_audit _configtracker _dsappevent _dsclient _dsphonehome _internal _introspection _telemetry _thefishbucket	_audit _configtracker _dsappevent _dsclient _dsphonehome _internal _introspection _telemetry _thefishbucket

Utility SPL - /search/jobs Summary Population

SPL for collecting /search/jobs information to write to summary index. Schedule frequently, every ~2/3min ideally

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
   search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| rename title AS searchQuery, sid AS search_id, label AS savedsearch_label, eai:acl.app AS appName, eai:acl.owner AS owner, eventFieldCount AS rawEventFieldsReturned
   ``` FILTER - Drop any search jobs you know you don't want - Defaults are any search/jobs query and either of the presentation collection queries ```
| search NOT searchQuery IN ("*/search/jobs*", "*search_log_events*", "*search_jobs_rest_collection*", "*search_jobs_search_log_data*")
   ``` FILTER - Drop any SIDs that have already been written to the summary location
   Subsearch passes compiled filter string: splunk_server=<host> AND search_id IN (...)```
| search NOT [search index=summary sourcetype=search_jobs_rest_collection earliest=-4h@h latest=+4h@h
   | rex field=_raw ", search_id=\"?(?<search_id>[^\",]+)\"?,"
   | rex field=_raw "orig_splunk_server=\"(?<orig_splunk_server>[^\"]+)"
   | stats values(search_id) AS search_id_list BY orig_splunk_server
   | eval "Combined Subsearch Filter for REST API Collection Job" = "( splunk_server=\"" . orig_splunk_server . "\" AND search_id IN (\"" . mvjoin(search_id_list, "\", \"") . "\") )"
   | rename "Combined Subsearch Filter for REST API Collection Job" AS search
   | table search]
   ``` FILTER - to search artifacts published within last 4h - Timerange here should be same as in the search NOT filter above/below. ```
| eval _time = strptime(published, "%Y-%m-%dT%H:%M:%S.%3N")
| where _time >= relative_time(now(), "-4h")
   ``` Output to summary index will not include original SPL, if you want the original SPL join on sourcetype=audittrail with search_id ```
| table _time, splunk_server, search_id, savedsearch_label, appName, owner, dispatchState, keywords, runDuration, searchEarliestTime, searchLatestTime, diskUsage, rawEventFieldsReturned, resultCount, eventCount, scanCount, dropCount,
searchTotalEliminatedBucketsCount, searchTotalBucketsCount, performance.command.search.index.invocations, performance.command.search.kv.duration_secs, priority, provenance, dispatchAs, request.ui_dispatch_app, eventSearch, published
   ``` Handle all empty string false-null fields returned from API by explicitly setting to null if length is less than 1 ```
| foreach * [| eval <<FIELD>> = if( isnull('<<FIELD>>') OR len('<<FIELD>>') < 1, null(), '<<FIELD>>')]
   ``` Generate normalized SID to populate savedsearch_label value properly across search artifacts ```
| `get_normalized_search_id(search_id)`
   ``` Aggregate key fields across SIDs using normalized SID - Note: min_sid_length is used to identify primary search ```
| eventstats min(eval(len(search_id))) AS min_sid_length, values(savedsearch_label) AS savedsearch_label, values(keywords) AS all_keywords BY search_id_normalized
| eval isPrimarySID = if( len(search_id)=min_sid_length AND !match(search_id, "^(remote|subsearch)"), 1, 0 )
| fields - min_sid_length
   ``` FILTER - Do not write summary index event if the search is not DONE - wait until this point to ensure label is passed to all subsearch artifacts ```
| search dispatchState="DONE"
   ``` Clean/Modify data as needed in preparation to write to summary index ```
| eval endpoint = "/servicesNS/-/-/search/jobs"
| foreach searchEarliestTime, searchLatestTime, runDuration [| eval <<FIELD>> = round('<<FIELD>>', 0)]
| addinfo | fields - info_*time | rename info_sid AS population_sid
   ``` Write output from search/jobs base collection to Summary Index - use appendpipe to avoid writing summary events in ad-hoc search ```
| appendpipe
   [| where match(population_sid, "^(scheduler|_scheduler_)")
   | collect testmode=f addinfo=f index=summary sourcetype=search_jobs_rest_collection marker="search_jobs_rest_collection_job"
   | where false()]
    ``` END search/jobs collection - BEGIN search.log collection ```
```


https://docs.splunk.com/Documentation/Splunk/latest/RESTREF/RESTsearch#search.2Fjobs
```

Utility SPL - search.log Summary Population

SPL for collecting search.log information to write to summary index. Schedule frequently, every ~2/3min ideally.

NOTE: EXPECTS OUTPUT FROM /search/jobs Summary Population in this PDF!

```
[...]   ``` END search/jobs collection - BEGIN search.log collection ```
   ``` FILTER - Only pass the primary SIDs ```
| search isPrimarySID = 1
   ``` search.log can be split into multiple files for larger searches - here we generate new rows for each search ID to ensure we capture these expanded log files if they exist.
       NOTE: This will generate many search message errors since most of these will not exist most of the time. ```
| eval search_log_counter = split("search.log, search.log.1, search.log.2", ", ")
| mvexpand search_log_counter
   ``` Set the REST url the map command will query by using the search ID and the generated counter reference ```
| eval search_job_rest_url = "/servicesNS/-/-/search/jobs/" . search_id . "/" . search_log_counter
   ``` Run a new search to collect search.log for each input row, specifying the splunk_server that returned data previously. Pass the input field values by eval'ing them within map ```
| map maxsearches=200000
   search="| rest \"$search_job_rest_url$\" splunk_server=\"$splunk_server$\"
   | eval search_id = \"$search_id$\" | eval endpoint = \".../$search_log_counter$\" | eval app = \"$appName$\" | eval owner = \"$owner$\"
   | eval savedsearch_label = \"$savedsearch_label$\"
   | rename value AS search_log_line
   "
   ``` Convert search.log text block into multivalue, then split into individual rows ```
| makemv search_log_line tokenizer="([^\n]+)"
| stats first(*) AS * BY search_id, search_log_line
    ``` FILTER - Only keep search.log lines that we care about ```
| search search_log_line IN ("*BatchSearch is initialized*", "*Search requires the following indexes*", "*IndexScopedsearch is called for index*")
   ``` Clean/Modify search.log data as needed in preparation to write to summary index.
   NOTE: search_log_events raw format follows schema: <original search.log line> ~~~META:, <meta fields from collection job> ```
| rename splunk_server AS log_src_server
| eval _raw = search_log_line . " ~~~META:"
   ``` Manually create the _raw for the summary event, concatenating the original search.log line and the other fields from collection job in csv key=value format
  NOTE: This ensures a consistent ordering of fields. Do not change the order of the foreach field references without updating the regular expressions in search_id extractions as well. ```
| foreach app, endpoint, search_id, log_src_server, owner, savedsearch_label
   [| eval _raw = if( isnotnull('<<FIELD>>'), _raw . ", <<FIELD>>=\"" . '<<FIELD>>' . "\"", _raw)]
   ``` Write output of search.log collection to Summary Index - use appendpipe to avoid writing summary events in ad-hoc search ```
| appendpipe
   [| addinfo | fields - info_*time | rename info_sid AS population_sid
   | where match(population_sid, "^(scheduler|_scheduler_)") | fields - population_sid
   | collect testmode=f addinfo=f index=summary sourcetype=search_jobs_search_log_events marker="search_jobs_rest_collection_search_log_data"
   | where false()]
```
```

Utility SPL - search.log Summary Population

Full SPL for collecting both search/jobs base and search.log data, had to be tiny to fit every line. Schedule for every 2 minutes

```
| rest /servicesNS/-/-/search/jobs splunk_server=*
   search="sid!=SummaryDirector*" search="sid!=rsa_*" search="sid!=rt_*"
| rename title AS searchQuery, sid AS search_id, label AS savedsearch_label, eai:acl.app AS appName, eai:acl.owner AS owner, eventFieldCount AS rawEventFieldsReturned
   ``` FILTER - Drop any search jobs you know you don't want - Defaults are any search/jobs query and either of the presentation collection queries ```
| search NOT searchQuery IN ("*/search/jobs*", "*search_log_events*", "*search_jobs_rest_collection*", "*search_jobs_search_log_data*")
   ``` FILTER - Drop any SIDs that have already been written to the summary location
   Subsearch passes compiled filter string: splunk_server=<host> AND search_id IN (...)```
| search NOT [search index=summary sourcetype=search_jobs_rest_collection earliest=-4h@h latest=+4h@h
   | rex field=_raw ", search_id=\"?(?<search_id>[^\",]+)\"?," | rex field=_raw "orig_splunk_server=\"(?<orig_splunk_server>[^\"]+)"
   | stats values(search_id) AS search_id_list BY orig_splunk_server
   | eval "Combined Subsearch Filter for REST API Collection Job" = "( splunk_server=\"" . orig_splunk_server . "\" AND search_id IN (\"" . mvjoin(search_id_list, "\", \"") . "\") )"
   | rename "Combined Subsearch Filter for REST API Collection Job" AS search | table search]
   ``` FILTER - to search artifacts published within last 4h - Timerange here should be same as in the search NOT filter above/below. ```
| eval _time = strptime(published, "%Y-%m-%dT%H:%M:%S.%3N")
| where _time >= relative_time(now(), "-4h")
   ``` Output to summary index will not include original SPL, if you want the original SPL join on sourcetype=audittrail with search_id ```
| table _time, splunk_server, search_id, savedsearch_label, appName, owner, dispatchState, keywords, runDuration, searchEarliestTime, searchLatestTime, diskUsage, rawEventFieldsReturned, resultCount, eventCount, scanCount, dropCount, searchTotalEliminatedBucketsCount, searchTotalBucketsCount, performance.command.search.index.invocations,
performance.command.search.kv.duration_secs, priority, provenance, dispatchAs, request.ui_dispatch_app, eventSearch, published
   ``` Handle all empty string false-null fields returned from API by explicitly setting to null if length is less than 1 ```
| foreach * [| eval <<FIELD>> = if( isnull('<<FIELD>>') OR len('<<FIELD>>') < 1, null(), '<<FIELD>>')]
   ``` Generate normalized SID to populate savedsearch_label value properly across search artifacts ```
| `get_normalized_search_id(search_id)`
   ``` Aggregate key fields across SIDs using normalized SID - Note: min_sid_length is used to identify primary search ```
| eventstats min(eval(len(search_id))) AS min_sid_length, values(savedsearch_label) AS savedsearch_label, values(keywords) AS all_keywords BY search_id_normalized
| eval isPrimarySID = if( len(search_id)=min_sid_length AND !match(search_id, "^(remote|subsearch)"), 1, 0 )
| fields - min_sid_length
   ``` FILTER - Do not write summary index event if the search is not DONE - wait until this point to ensure label is passed to all subsearch artifacts ```
| search dispatchState="DONE"
   ``` Clean/Modify data as needed in preparation to write to summary index ```
| eval endpoint = "/servicesNS/-/-/search/jobs"
| foreach searchEarliestTime, searchLatestTime, runDuration
   [| eval <<FIELD>> = round('<<FIELD>>', 0)]
| addinfo | fields - info_*time | rename info_sid AS population_sid
   ``` Write output from search/jobs base collection to Summary Index - use appendpipe to avoid writing summary events in ad-hoc search ```
| appendpipe
   [| where match(population_sid, "^(scheduler|_scheduler_)")
   | collect testmode=f addinfo=f index=summary sourcetype=search_jobs_rest_collection marker="search_jobs_rest_collection_job"
   | where false()]
   ``` END search/jobs collection - BEGIN search.log collection ```
   ``` FILTER - Only pass the primary SIDs ```
| search isPrimarySID = 1
   ``` search.log can be split into multiple files for larger searches - here we generate new rows for each search ID to ensure we capture these expanded log files if they exist.
       NOTE: This will generate many search message errors since most of these will not exist most of the time. ```
| eval search_log_counter = split("search.log, search.log.1, search.log.2", ", ")  | mvexpand search_log_counter
   ``` Set the REST url the map command will query by using the search ID and the generated counter reference ```
| eval search_job_rest_url = "/servicesNS/-/-/search/jobs/" . search_id . "/" . search_log_counter
   ``` Run a new search to collect search.log for each input row, specifying the splunk_server that returned data previously. Pass the input field values by eval'ing them within map ```
| map maxsearches=200000
   search="| rest \"$search_job_rest_url$\" splunk_server=\"$splunk_server$\" | eval search_id = \"$search_id$\" | eval endpoint = \".../$search_log_counter$\" | eval app = \"$appName$\" | eval owner = \"$owner$\" | eval savedsearch_label = \"$savedsearch_label$\"
   | rename value AS search_log_line"
   ``` Convert search.log text block into multivalue, then split into individual rows ```
| makemv search_log_line tokenizer="([^\n]+)"
| stats first(*) AS * BY search_id, search_log_line
   ``` FILTER - Only keep search.log lines that we care about ```
| search search_log_line IN ("*BatchSearch is initialized*", "*Search requires the following indexes*", "*IndexScopedsearch is called for index*")
   ``` Clean/Modify search.log data as needed in preparation to write to summary index.
   NOTE: search_log_events raw format follows schema: <original search.log line> ~~~META:, <meta fields from collection job> ```
| rename splunk_server AS log_src_server
| eval _raw = search_log_line . " ~~~META:"
   ``` Manually create the _raw for the summary event, concatenating the original search.log line and the other fields from collection job in csv key=value format
   NOTE: This ensures a consistent ordering of fields. Do not change the order of the foreach field references without updating the regular expressions in search_id extractions as well. ```
| foreach app, endpoint, search_id, log_src_server, owner, savedsearch_label
   [| eval _raw = if( isnotnull('<<FIELD>>'), _raw . ", <<FIELD>>=\"" . '<<FIELD>>' . "\"", _raw)]
   ``` Write output of search.log collection to Summary Index - use appendpipe to avoid writing summary events in ad-hoc search ```
| appendpipe
   [| addinfo | fields - info_*time | rename info_sid AS population_sid
   | where match(population_sid, "^(scheduler|_scheduler_)") | fields - population_sid
   | collect testmode=f addinfo=f index=summary sourcetype=search_jobs_search_log_events marker="search_jobs_rest_collection_search_log_data"
   | where false()]
```

We made it!

.conf24

splunk>

Recap

What we covered today

- Introduction and framing of primary default Splunk indexes for search data
- Technical Walkthrough of:
  - Examples of using default indexes to identify problematic searches
  - Ways to unify search data across multiple data sources for enrichment.
  - REST Endpoints for search data - */search/jobs*
- Review of REST-based collection strategies available to all customers.
- Approach to gathering and storing search data via SPL

Key Takeaways

- Default Splunk data sources can tell us a lot about our searches,
  but nothing has a "complete" view of data source usage
- Gaining insights can be complex and require multiple sources.
  Leverage the tooling and utilities already available where you can.
- Keep an eye out for updates to audit logging in future Splunk releases!

https://github.com/TheWoodRanger/presentation-conf_24_audittrail_native_telemetry

Call to Action

- Grab the resources off Github for additional reporting usecases & references
  GitHub:
  https://github.com/TheWoodRanger/presentation-conf_24_audittrail_native_telemetry

- Check out the apps/utilities/tools referenced in slides
  - Github will be updated with any new ones we find out about

- Use the references & resources to great presentations

- Reach out!
  - Splunk UserGroups Slack: TheWoodRanger / richgalloway

© 2024 SPLUNK INC.

Questions?

Resources

Ryan's 2023 Talk

- PLA1881C - Maximizing Splunk SPL™: Foreach and the Power of Iterative, Templatized Evals

Ryan's Bsides SPL 2019 Presentation

- What's in my Data? Field Analysis for the Advanced Engineer

Tips on managing scheduled searches

- Scheduled Search Management - white paper by David Paper

Redundant & Inefficient Search Spotter app - PLA1457B (.conf24)

More Resources

Term Efficiency, Search Optimization

- PLA1089C - TSTATS and PREFIX, How to get the most out of your lexicon with walklex, tstats, indexed fields, PREFIX, TERM
- PLA1466B - Fields, Indexed Tokens, and You
- PLA1258C - I Am Speed! Searching on Your Own TERMs With Simple Techniques That 99% Aren’t Using!
- TRU1133B - Clara-Fication: More Tstats for Your Buckets
- Splunk Blog - Splunk Clara-fication Search Best Practices
- Tips from Splunk Docs to Write better searches
- Splunk Blog: Splunk Clara-fication Dashboarding Best Practices
- Dashboard Studio resources: Improving dashboard performance and resource usage (15 minutes), Dashboard Studio

More Resources 2

On the topic of understanding job performance:

- TRU1143C - Clara-fication: Job Inspector
- PLA1162B - Clara-Fication: Finding and Improving Expensive Searches
- YouTube - Using the Splunk job inspector
- Splunk Concurrency Helper - https://github.com/nicovdw/splunk_concurrency_helper
- Getting Smarter about Splunk SmartStore - https://github.com/redvelociraptor/gettingsmarter/tree/main

More Resources 3

On the topic of Security Usecase/SPL Techniques:

- PLA1528B - Master Joining Datasets Without Using Join
- PLA1261B - Beyond REGULAR Regular Expressions v3.0
- PLA1547B - Lighter, Faster and Calmer Ways to Learn Splunk® Enterprise With | makeresults, | gentimes and Some Random()% Too!
- TRU1192B - Getting To Know Your Data
- Turning Security Use Cases into SPL
- How to compare fields over multiple sourcetypes without 'join', 'append' or use of subsearches?
- Security Ninjutsu Series - also available via David Veuve's website
  - Security Ninjutsu Part Four
  - Security Ninjutsu Part Five
  - Security Ninjutsu Part Six
- Use _internal to monitor hosts - https://github.com/silkyrich/cluster_health_tools/
- Dashboards for monitoring Splunk health - https://github.com/dpaper-splunk/public/tree/master/dashboards

Yet Another Resource Page

Check these out, too

Use _internal to monitor hosts

https://github.com/silkyrich/cluster_health_tools/

Dashboards for monitoring Splunk health

https://github.com/dpaper-splunk/public/tree/master/dashboards

Dashboards to assist with optimising concurrency settings

https://github.com/nicovdw/splunk_concurrency_helper

Getting Smarter about Splunk SmartStore

https://github.com/redvelociraptor/gettingsmarter/tree/main

Thank you

Appendix

Cool stuff we didn’t have time to talk about

Disclaimers

Feel free to use the SPL in these slides, but test it before using it in a production environment.

Use at your own risk - it's difficult to test for every environment possibility. Your results may vary.

The SPL is intended to show what is possible and may not be efficient. Nor is it exhaustive.

The searches that extract information from dashboard code were not tested against SPL2 queries.

Reach out if you find problems - ryan.wood@guidepointsecurity.com

_internal index

Notes about some components

component = Metrics

Caution: Values are based on sampled data so they may be incomplete

component = DC:DeploymentClient

Can help find Deployment Server clients

component = TCPInputProc

Can help find TCP inputs. Watch out for inputs that stop reporting.

sourcetype = splunkd_remote_searches

Most stats here apply only to the indexers; don’t forget to add CPU time used by SH

Interesting numbers to watch: index_bucketcache_miss_wait, rawdata_bucketcache_miss_wait

-Rich

Lookups used by searches

Why bother?

- Know which lookup tables are being used in a search
- Clean up unused lookups
- Reduce the size of the knowledge bundle

-Rich

Lookups used by searches

What the data looks like

Savedsearch Name	Query
CloudTrail Base Search	`aws-cloudtrail((aws_account_id="*"), (region="**") )` | lookup unauthorized_errorCode errorCode OUTPUT Unauthorized
host_role_generator	index="summary" source="splunk-svc-consumer" | stats values(search_head_names) as hrole by usage_source | rename usage_source as host | outputlookup hostrole.csv
Metrics Selectable Lookup	| inputlookup dropdowns.csv | stats values(host) as host by unix_category unix_group
splunk_identities_custom_gap_report	| from lookup:splunk_rest_identities_kv_store_lookup | search NOT [| inputlookup splunk_identities_custom_kv_store_lookup | fields identity] | table …

-Rich

Lookups used by searches

How?

```
| rest splunk_server=local /servicesNS/-/-/saved/searches ``` Fetch scheduled searches ```
| fields title eai:acl.app search
| append [| rest splunk_server=local /servicesNS/-/-/data/ui/views ``` Fetch dashboards ```
   | fields title eai:acl.app eai:data | rename eai:data as search ]
``` Look for queries that use lookup commands ```
| regex search="\|\s*(?:inputlookup|lookup|outputlookup|from\s+lookup:)"
``` Extract the lookup name ```
| rex field=search max_match=0
"\|\s*(?:inputlookup|lookup|outputlookup|from\s+lookup:)\s+(?:append\s*=\s*\w+\s+)?(?<lookup>[\w\.]+)"
``` List the lookups used by each app ```
| stats values(lookup) as lookups by eai:acl.app
``` Optional - Show distinct lookup names ```
| fields lookups | mvexpand lookups | dedup lookups | sort + lookups
```

-Rich

Datamodels used by searches

Why worry?

- Unused datamodels don’t need to be accelerated, saving search slots and SVCs
- Know which datamodels should remain accelerated

-Rich

Datamodels used by searches

What the data looks like

Search/Dashboard Name	Query
Critical Severity Intrusion	| tstats local=false summariesonly=true allow_old_summaries=true count from datamodel=Intrusion_Detection.IDS_Attacks where …
DA-ITSI-LB-Upstream_Members	| datamodel LoadBalancer Connections search | search host="$host$"
compliance	<form … <query>| from datamodel:"Malware"."Malware_Operations" …

-Rich

Datamodels used by searches

How to do it

Some events that we don’t handle include:

- DM name in a token (as in a dashboard or the ‘map’ command) is ignored
- The ‘datamodel’ command with no arguments (which displays all DMs) is ignored
- Commented ‘datamodel’, ‘from datamodel’, or ‘tstats… from datamodel’ commands are treated as un-commented

```
| rest splunk_server=local /servicesNS/-/-/saved/searches  ``` Fetch scheduled searches ```
| fields title eai:acl.app search
| append [| rest splunk_server=local /servicesNS/-/-/data/ui/views ``` Fetch dashboards ```
| fields title eai:acl.app eai:data | rename eai:data as search ]
``` We only care about searches that use the ‘datamodel’ command, the ‘from datamodel’, or the ‘from datamodel’ option to the ‘tstats’  command ```
| regex search="\|\s*datamodel[\s=]"
``` Extract the DM name ```
| rex field=search max_match=0 "(?:\|\s*|\s+from\s+)datamodel[\s=:]\\\"?(?<dm>[\w]+)\\\"?"
``` Ignore events we couldn’t get a DM name from ```
| where isnotnull(dm)
``` List the DM names by app ```
| stats values(dm) as DMs by eai:acl.app
| table eai:acl.app DMs
| rename eai:acl.app as App, DMs as DataModels
```

-Rich

Unused Sourcetypes

Why do we care?

- If it isn’t used then maybe it doesn’t have to be ingested
- If it isn’t used and has to be ingested then maybe it can be stored elsewhere
  - Ingest Action to S3

-Rich

Unused Sourcetypes

More info

The results of this query are meant to be used to start talks with your users. Find out if they're really not using the sourcetype or if it's one that is needed only under certain circumstances (break glass, audit, etc.). It's important to note that the output of this search improves as the time range is extended.

Please do not feed them into automation that deletes inputs or indexes

```
index=_audit sourcetype=audittrail info=completed action=search
``` Extract sourcetype names from the audittrail events ```
| rex max_match=0 "sourcetype_count__(?<st>\w+)=(?<cnt>\d+)"
| fields st cnt
``` Discard events without a sourcetype ```
| where isnotnull(st)
| append
   ``` Get a list of all sourcetypes ```
   [| tstats count where index=* by sourcetype,index | stats values(index) as indexes by sourcetype
   | rename sourcetype as st | eval cnt=0]
``` Ignore Splunk default sourcetypes (not an exhaustive list) ```
| search NOT st IN (audittrail http_event_collector_metrics itsi_internal_log scheduler splunk*)
``` Count how many times each sourcetype is used ```
| stats sum(cnt) as cnt, values(indexes) as indexes by st
``` Keep only the sourcetypes with count of zero ```
| where cnt=0
| fields - cnt
| rename st as sourcetype
```

-Rich

Find Inactive Users

```
| rest splunk_server=local /servicesNS/-/-/admin/users
| fields title realname last_successful_login
| eval lastLogin = if(isnull(last_successful_login) OR last_successful_login=0, "never",
strftime(last_successful_login, "%c"))
| eval idleDays = round((now()-last_successful_login)/86400,0)
| where (idleDays > 90 OR lastLogin = "never")
| table title realname lastLogin idleDays
| rename title as User, realname as Name, lastLogin as "Last Login Time", idleDays as "Days Since Last Login"
```

-Rich

Objects owned by inactive users

```
| rest splunk_server=local /servicesNS/-/-/admin/directory | fields title eai:acl.app eai:acl.owner eai:type
eai:acl.sharing
| rename eai:acl.* as *
| where (owner!="nobody" AND owner!="admin")
| search [| rest splunk_server = local /servicesNS/-/-/admin/users
| fields title realname last_successful_login
| eval lastLogin = if(isnull(last_successful_login) OR last_successful_login=0,"never",
strftime(last_successful_login, "%c"))
| eval idleDays = round((now()-last_successful_login)/86400,0)
| where (idleDays > 90 OR lastLogin = "never")
| fields title
| rename title as owner | format]
```

-Rich

Unused Indexes

Why do we care?

- Don’t pay to store data we don’t use
- Less to manage
- Faster indexer startup

If the data is needed, consider using Ingest Actions to send to S3 instead of an index.

-Rich

Unused Indexes

How to do it?

Get a list of unused indexes and subtract that from a list of all indexes.

- Extract index names from searches, via audit logs or saved search queries
- Expand macros to see if the contain index names
- Expand tags and eventtypes to see if they contain index names
- If “index=*” is found, fetch the names of all indexes that user can access
- If no index name is found, fetch the user’s default indexes

OR

- Index all search.log files
- Extract index names from ‘INFO IndexScopedSearch [22436 localCollectorThread] - IndexScopedSearch is called for index = history’ events.
- Know this doesn’t work in Splunk Cloud

-Rich

Unused Indexes

A possible solution

```
| rest /services/data/indexes splunk_server=local
| search title!="_*" NOT
[ search index=_internal sourcetype=splunkd_remote_searches source=*remote_searches.log "Streamed search connection closed:"
| regex search="^(litsearch|mcatalog|mstats|mlitsearch|litmstats|tstats|presummarize)" ```these lines have search expanded```
| rex field=search max_match=0 "(?s)\|?\s*(mlitsearch)\s+.*?\[(?P<subsearch>.*?)\]\s*(\||$)" ```retrieving subsearch(es)```
| rex field=search "(?s)(?P<prepipe>\s*\|?([^\|]+))" ```retrieving prepipe```
| nomv subsearch ```Converts subsearch from multivalue to single value with values separated by linefeed```
| fillnull value="" subsearch ```eliminate null values of subsearch```
| eval search = prepipe . " " . subsearch ```Concatenates prepipe search with its subsearch(es)```
| rex field=search "(?s)(NOT\s+index(\s*=\s*|::)[^ ]+)|(NOT\s+\([^\)]+\))|(index(\s*=\s*|::)\"?(?P<indexregex1>[\*A-Za-z0-9-_]+))"
max_match=50 ```gets value(s) after "index=" or "index::" into 'indexregex'```
| rex field=search "(?s)(NOT\s+index(\s*IN\s*)[^ ]+)|(NOT\s+\([^\)]+\))|(index(\s*IN\s*\()(?P<indexregex2>[^\)]+))" max_match=50 ```gets
indexes specified in 'IN' statement```
| eval indexregex2=trim(split(indexregex2,","),"\s\""), index=mvappend(indexregex1,indexregex2) ```splits comma-delimited indexes from 'IN'
statement into mv field and trims quotes and spaces from name and appends indexes discovered from 'index=' or 'index::'```
| stats values(index) as index ```gets mv list of indexes attempted to search```
| mvexpand index ```places each unique, discovered index pattern on a separate row```
| where !match(index,"^_?\*+$") ```do not include index=* or index=_*```
| where !match(index,"^_\w+") ```do not include internal indexes``` | rename index as title | fields title | format]
| table title
```

Thanks to Tim Pacl of Splunk

-Rich