Tutorial:

Integrating Jamf Products with Splunk

Jamf Implementation Engineering
v2020-04-03ol

jamf

Table of Contents

Introduction

Purpose and Disclaimer

This document was prepared by Jamf's Implementation Engineering Group to help customers who want to use their Jamf data in Splunk and understand how the data available from Jamf Products can be combined to create actionable insights.

Nothing here is intended to imply that either Jamf or Splunk endorses the other's products.

Splunk and Jamf Products

Splunk is a data collection, indexing, reporting, and visualization solution that combines data feeds from many sources to create reports. Many of Splunk's customers use it to extract information from the large quantity of log files generated by modern IT operations. It might be used to support operational requirements like monitoring the average response time of a web site, or to support the forensics needs of compliance and IT security teams.

Jamf's products can participate in the Splunk ecosystem by contributing data on device configuration and management actions (Jamf Pro) and suspicious events occurring on Mac Computers (Jamf Protect).

Jamf Pro

Jamf Pro is a device management solution for Apple products, including Macs, iOS devices, and AppleTVs. Its Mac management framework includes an agent running on the endpoints to collect device configuration information. The management agent also runs policies like upgrading software applications and installing OS patches. Jamf Pro also has a Mobile Device Management (MDM) system to distribute device configuration profiles and send management actions like remote wipe/lock. These features can be combined to implement initial device configuration and ongoing management and monitoring.

The outcome of inventory and management actions are stored in Jamf Pro's database and can be retrieved via the Jamf Pro API. A free Splunk app that interfaces with the Jamf Pro API is available on Splunkbase. Additionally, Jamf Pro can be configured to send webhooks to Splunk when specific events occur for real-time monitoring and alerts.

Jamf Protect

Jamf Protect is a separate Jamf product that focuses on compliance monitoring and deep integration with macOS to provide event notifications that may be monitored and analyzed to detect problems. These events may be sent to a "Security Information and Event Management" (SIEM) system for aggregation and further analysis. Splunk is one of many solutions that can fill the SIEM role.

Combining Data Sources

Splunk can use the data provided by upstream systems to create Alerts, Dashboards, and to support forensic analysis. Alerts can be triggered when a threshold for a particular metric is exceeded (like the number of inbound connections attempted to a host) or when an unusual pattern is detected. Data can also be combined to create dashboards (groups of charts, graphs, heat maps, etc.) that can summarize

large amounts of data and provide a visual indication that something is changing. An important third use is threat detection and forensics. Security teams responding to an incident need to have large amounts of data at their fingertips for ad-hoc and often novel analysis.

Data aggregation allows the combination of data points from the multiple services that comprise an organization's IT ecosystem. For example, an analyst might require OS version data from Jamf Pro, user behavior data from Jamf Protect, and the access logs from a network proxy solution when researching a problem.

Creating Workflows

Splunk can trigger alerts based on criteria you select. When an alert is triggered, information about the event can be emailed, added to a report, or a webhook can be generated to trigger a response in another system. Some examples:

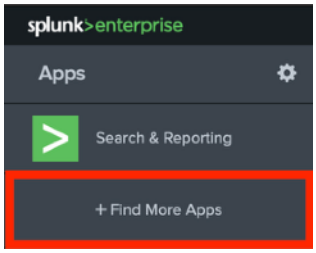
- If Jamf Protect reports an excessive number of screenshots being taken on a device, Splunk could trigger a workflow in your help desk software to create a security response incident to see if the user is aware of it and has a valid explanation or if some malware is responsible.
- If Jamf Pro reports the installation of a new software that involves an area of concern, integration with Splunk could be used to trigger a multi-tiered response:

  1. Jamf Pro itself can remove or block the execution of restricted software.
  2. Jamf Pro or Splunk can send webhooks to generate a quick-tip message to the user in the organization's chat app.
  3. A followup email could be sent to the user and their manager describing what was detected and referring the user to an internal web site that explains the area of concern in greater detail. For example, if the installation of an unauthorized BitTorrent client is detected, the email might direct the user to information about the organization's responsible-use policy. If they installed a file sharing app like DropBox, Box, or OneDrive in an organization that restricts such things, they might be sent information about data loss prevention and commitment to privacy.
  4. Some security teams might aggregate these events to develop indicators that user education on a larger issue is needed. For example, Splunk could add each event to a "Restricted Application Installation Attempts by Department" report to help target manager awareness-raising efforts and send quarterly reports via email.
  5. If a more serious condition occurs, raise a security incident.

Importing Jamf Pro Data into Splunk

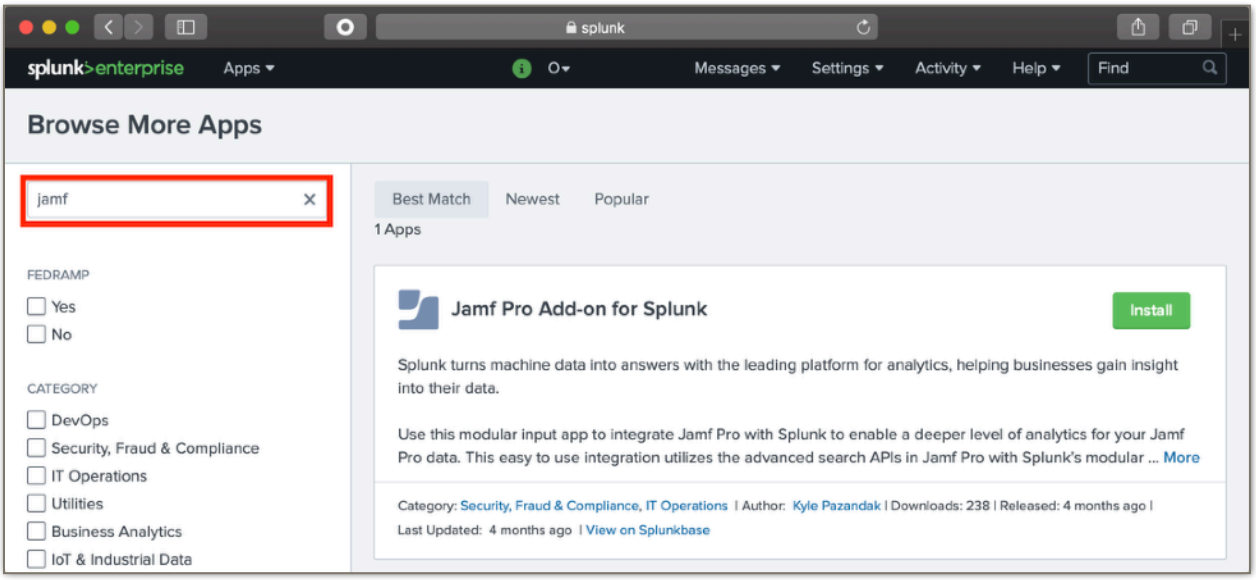
Install the Jamf Pro Add-on for Splunk

Jamf provides a free add-on for Splunk. It provides a simple means of retrieving device and management data from the Jamf Pro application and making it available in Splunk. You can obtain the add-on from SplunkBase, Splunk's integrated app store, or directly from within the Splunk app.



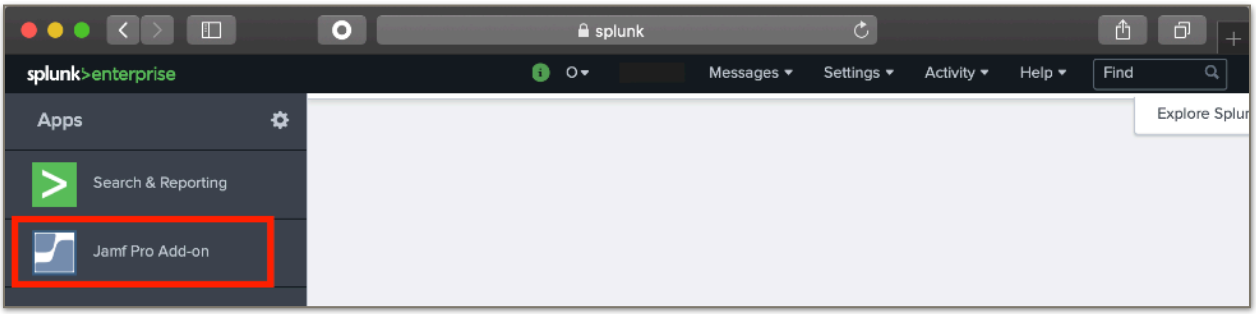
Log into your Splunk console. The currently-installed apps are listed on the left-hand side. At the end of the list, click the "Find More Apps" option....

Enter "Jamf" in the search box to locate the add-on and click the "Install" button.

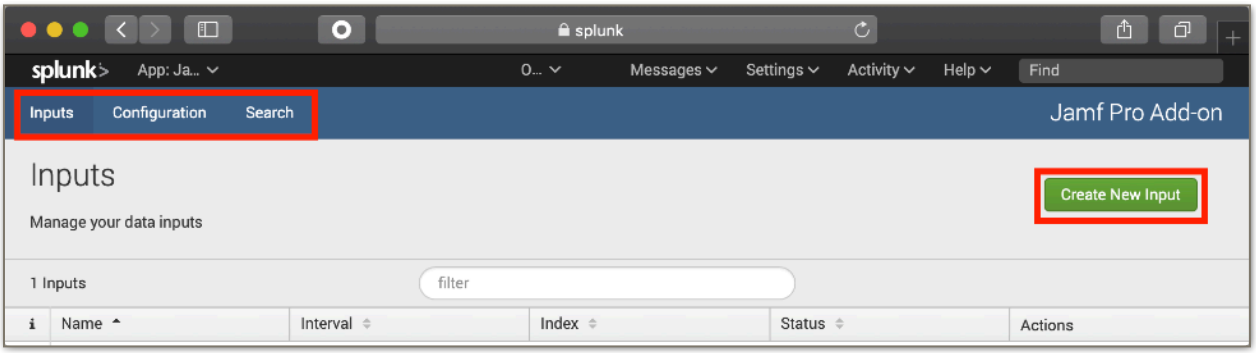


Configuring the Add-on

After the Add-on is installed, you'll see it in your app list. We'll need to click into the Add-on so we can enter some configuration information...

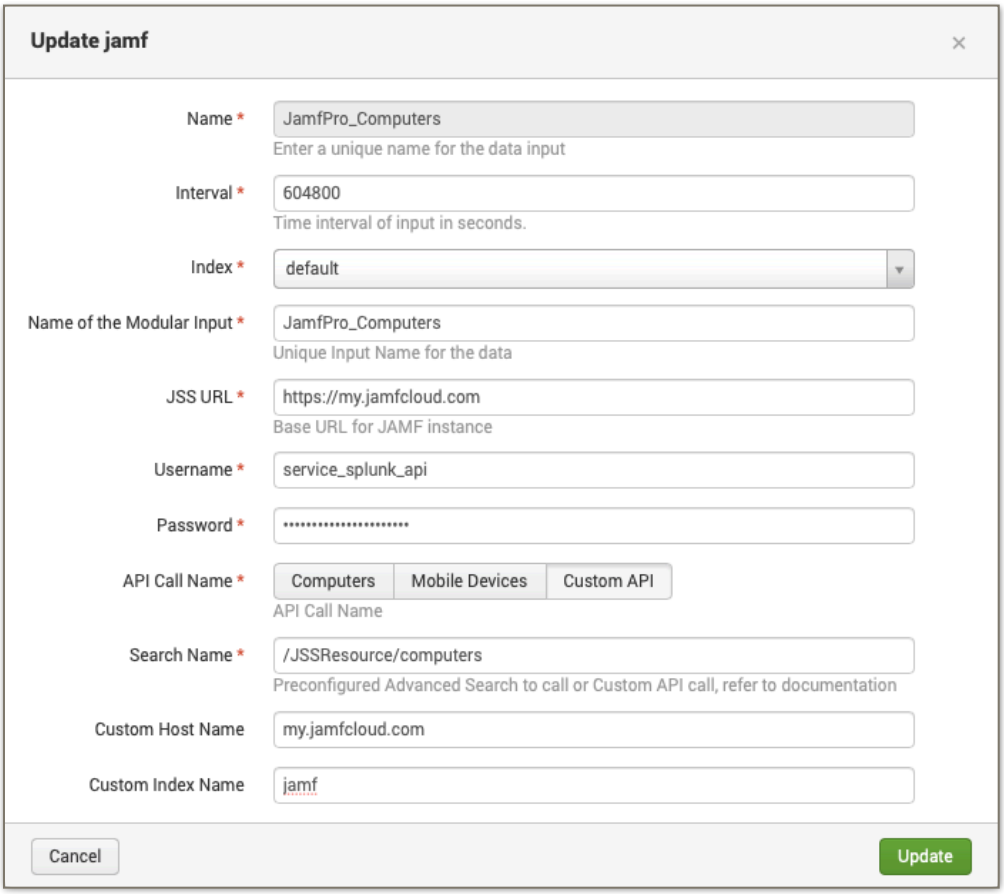


When you select the newly-installed app, you'll see three configuration categories at the top, "Inputs", "Configuration", and "Search". First, we'll use Inputs to tell the add-on how to contact our Jamf Pro instance. To begin, click the "Create New Input" button...



Each Input you create defines a Jamf Pro API endpoint to query, but to get started, you may want to look at the data provided by the **`/JSSResource/computers`** or **`/JSSResource/mobiledevices`** endpoints, depending on which device types you're managing.

In the example shown below, we're pulling data for "Custom API" /computers once per week, and have named the input "Jamf Pro". A custom host name is supplied so the data is attributed to our Jamf Pro...

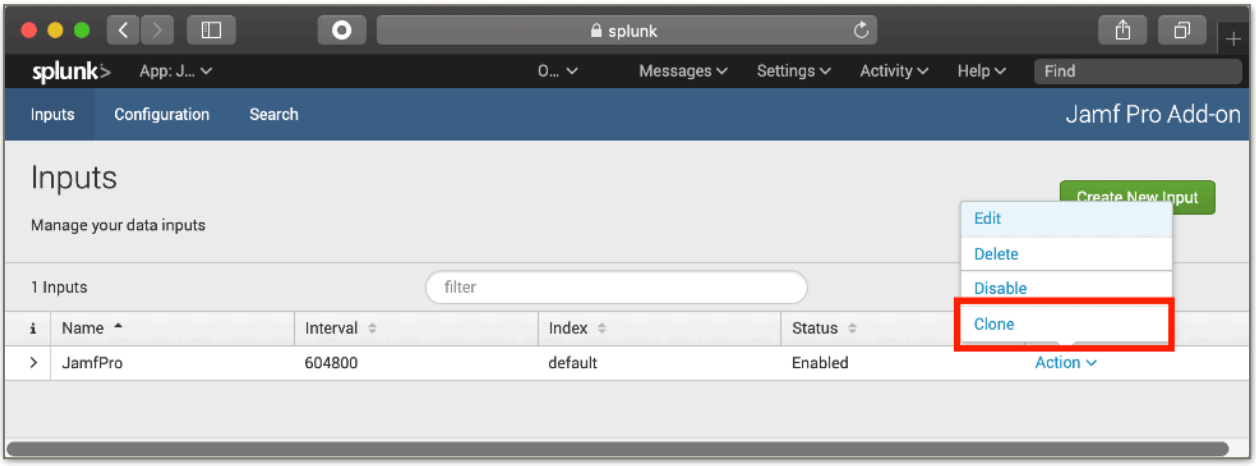


To configure the input, supply the following information:

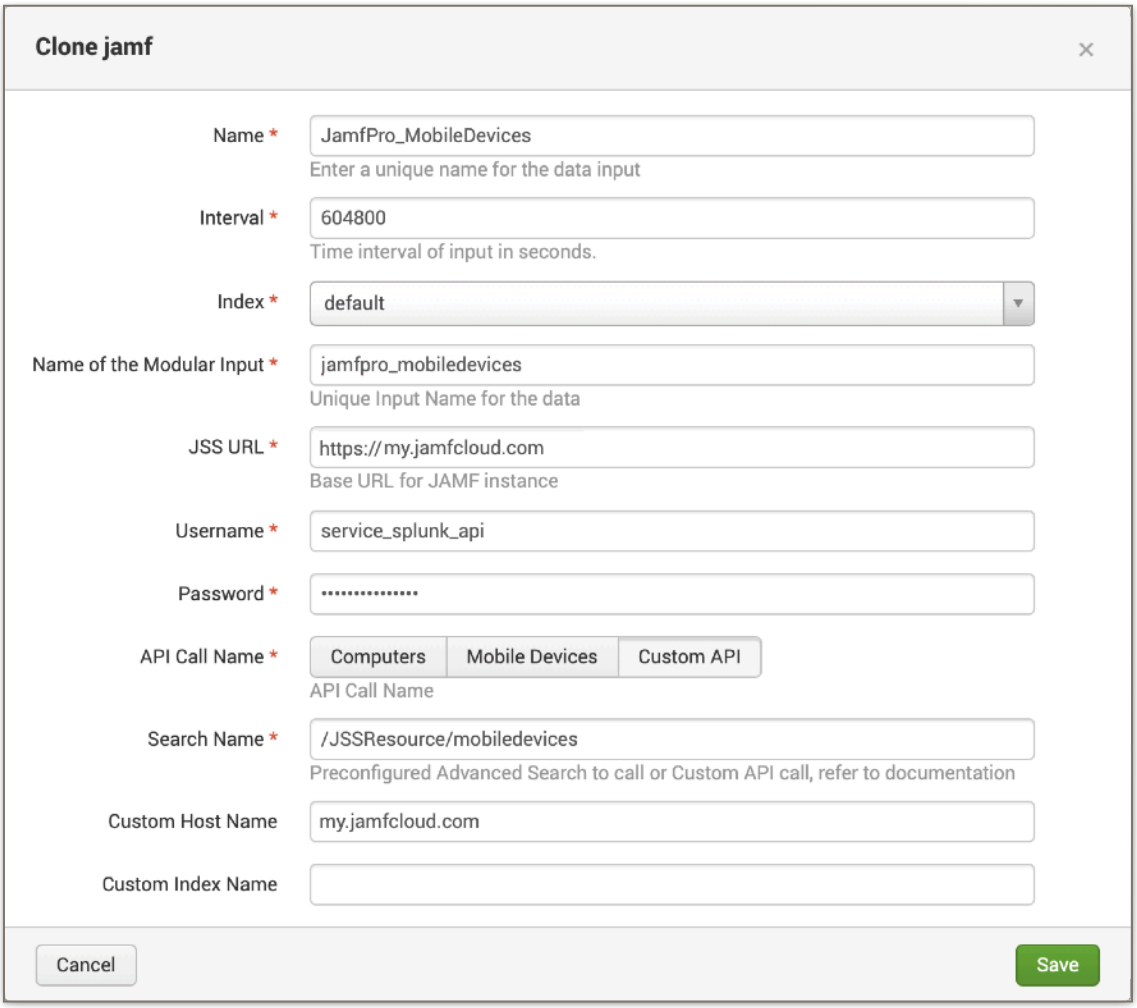
Name	A descriptive Name for the input, e.g. "*jamfpro_prod_computers*"
Interval	The refresh interval determines how frequently Splunk will import updated data. Daily=86400, weekly=604800, etc. Unless deleted by an administrator, Splunk will retain historical data to permit change detection and time-trend reporting.
Index	The Splunk index used for the data. Simple Splunk configurations use the default index (usually, "main"). But another index may be chosen if preferred.
Name of the Modular Input	Select a name for the input. Splunk searches can use this name to easily locate the data saved by this input.
JSS URL	The base URL for Jamf Pro, e.g. https://jamf.jamfcloud.com
Username	The Jamf Pro administrator will configure a service account in the Jamf Pro settings for use with API calls. For security, the Jamf admin should grant **read-only** permissions to the data that should be accessible to Splunk. Enter the service account's username here.
Password	Enter the password assigned to the Jamf Pro service account.
API Call Name	The type of Jamf Pro API call that will be made by this input. The Splunk add-in interfaces with Jamf's "Classic" API. The three options for specifying the API call type are Computers, Mobile Devices, or Custom API. The **Computers** and **Mobile Devices** options are used when you want Splunk to retrieve the output of a Jamf Pro "Advanced Search". Advanced Searches can be configured in Jamf Pro and allow easy setup of record filters ("criteria") and the fields that will be included in the report. Refs: Computers, Mobile Devices If you want to retrieve all records and all fields exposed by any other Jamf Pro API end-point, select the **Custom API** option.
Search Name	If **Computers** or **Mobile Devices** options are selected, this will be the name of the advanced search you want to run. When **Custom API** is used, search name will be the API endpoint you want to call. Many API endpoints representing different kinds of data are available in Jamf Pro. The most commonly used are `/JSSResource/computers` and `/JSSResource/mobiledevices`, which would pull complete details for these two Jamf Pro device classes. Alternately, many API end points allow data to be retrieved for a specific record. For example, you could specify `/JSSResource/computers/id/10` to pull data for only that device. Typically you would want data for all computers, but this feature might be useful when testing or when you want to have a more frequent data collection for particular devices that are being monitored closely. Similarly, some endpoints offer a "`/subset`" option. This is used to limit the amount of detail that will be returned.
Custom Host Name/Custom Index Name	Host and index are event metadata fields populated by Splunk when it receives data. Since the data is being pulled by the Splunk server, it will have that server's host name if custom host name is left blank. To fill this metadata with the name of the Jamf Pro server, fill in that server's host name instead. You can also enter custom values for these fields to make multiple inputs or multiple hosts appear as if they originated from a single source.

If needed, you can add additional inputs to import data from other Jamf Pro API endpoints. For example, if your initial entry was **`/JSSResource/computers`** but you also manage mobile devices (iPhones, iPads, and/or tvOS), you can add a second input for **`/JSSResource/mobiledevices.`**

To save the trouble of re-entering the server information, select Action > Clone on the first input...

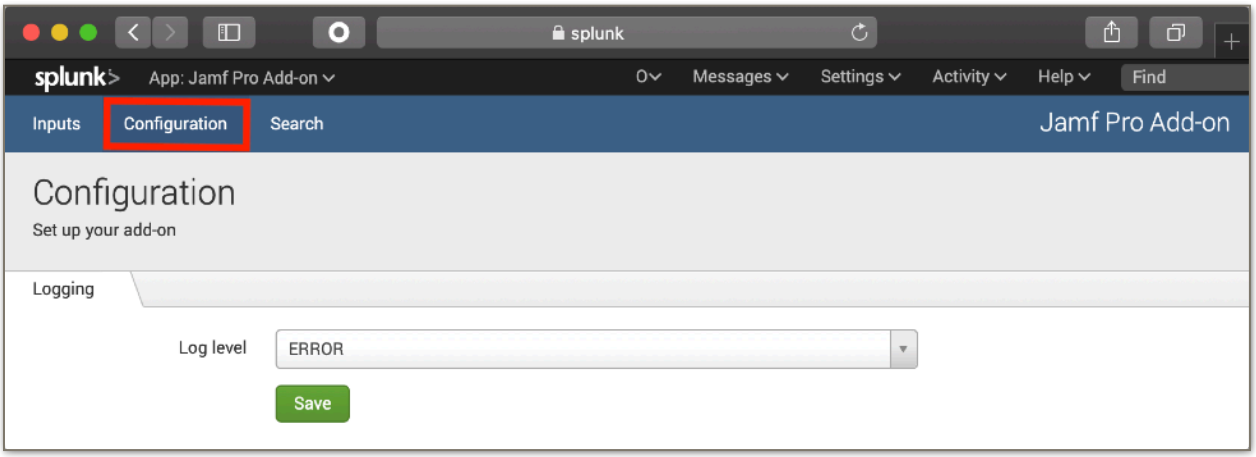


Enter a unique Name and Modular Input Name. Re-enter the Jamf Pro API Password, then click "Save"...



If you want to limit the amount of data flowing into your Splunk database, you can combine different endpoints and different frequencies to get the data you need when you need it. For example, you might pull the full **`/JSSResource/computers`** and **`/JSSResource/mobiledevices`** less frequently and use the selection criteria and reported-columns features of Jamf Pro Advanced Searches to pull just the data you want but on a more frequent basis. Different inputs with different retrieval frequencies can be combined to obtain an appropriate balance between data refresh period and database size based on your needs. A complete list of Jamf Pro's API end-points, query options, and example responses are available from https://developer.jamf.com/apis/classic-api/index.

You probably won't need to do anything in the Configuration or Search sections of the Jamf Pro Add-on. The only option in the Configuration" section is the log level. You can leave it set to error unless you need to troubleshoot the Add-on's behavior.



Troubleshooting:

If you aren't receiving data from Jamf Pro, double-check for typos in the the Jamf Pro URL, username, password, and api endpoint. Verify that your API user has read-only permissions on the Jamf Pro data for the endpoints you target. Some endpoints require permission on more than one data object. For example, the computers endpoint returns information about the users on a machine, so attempts to read it would fail if the API user didn't also have read permission on Users.

Default Log File Locations:

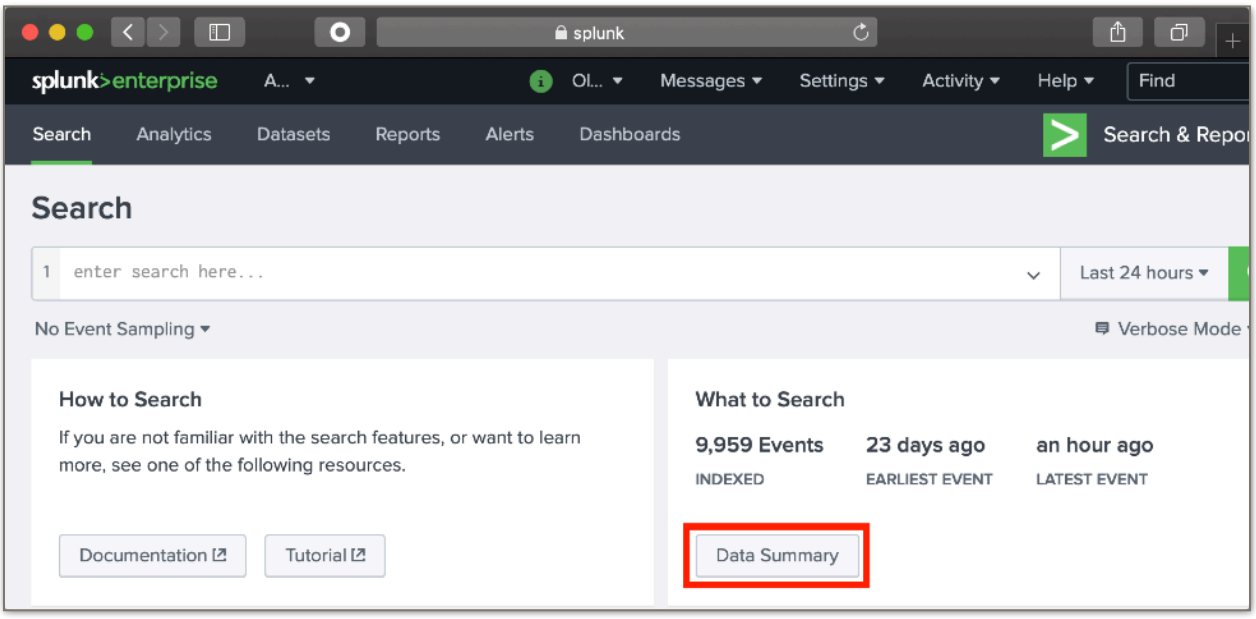
On Linux or Mac, the Jamf Add-on logs are written to:
`/opt/splunk/var/log/splunk/jamf_pro_addon_for_splunk_jamf.log`

On Windows, the Jamf Add-on logs are written to:
`\Program Files\splunk\var\log\splunk\jamf_pro_addon_for_splunk_jamf.log`

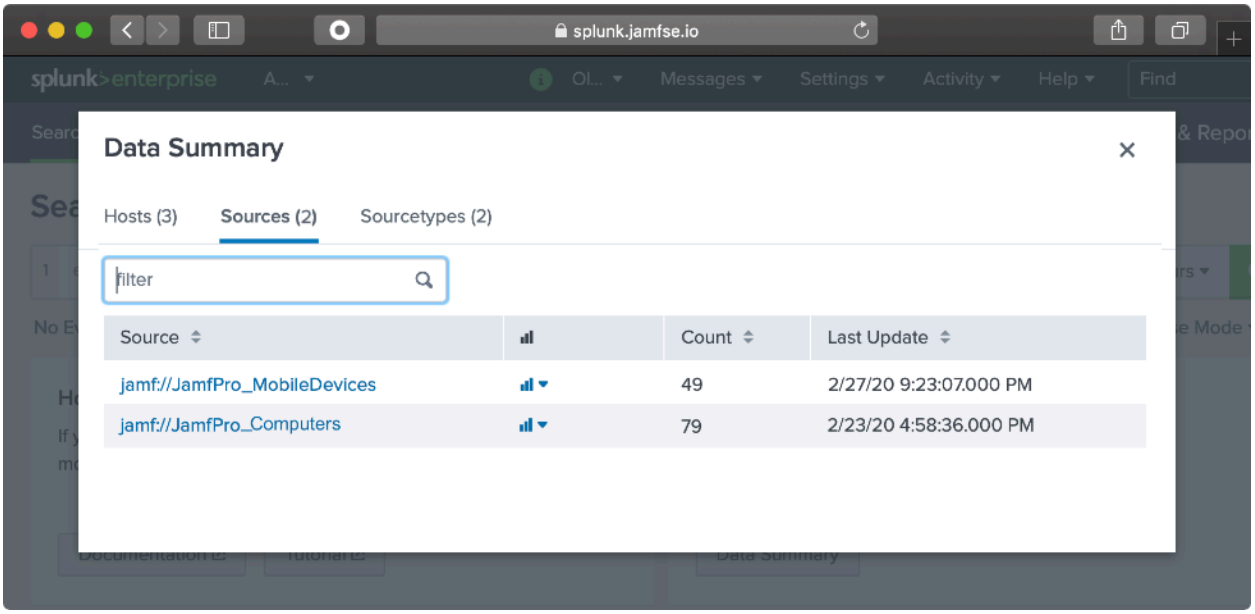
There is a third section called"Search". You can build searches and reports here if you want to limit their scope to the app, but often you'll do these things right in Splunk's standard "Search & Reporting" app.

Checking for data

The Jamf Pro add on will start retrieving data from the API as soon as you save your inputs. To take a look at the results, go to the Search & Reporting app and click "Data Summary".



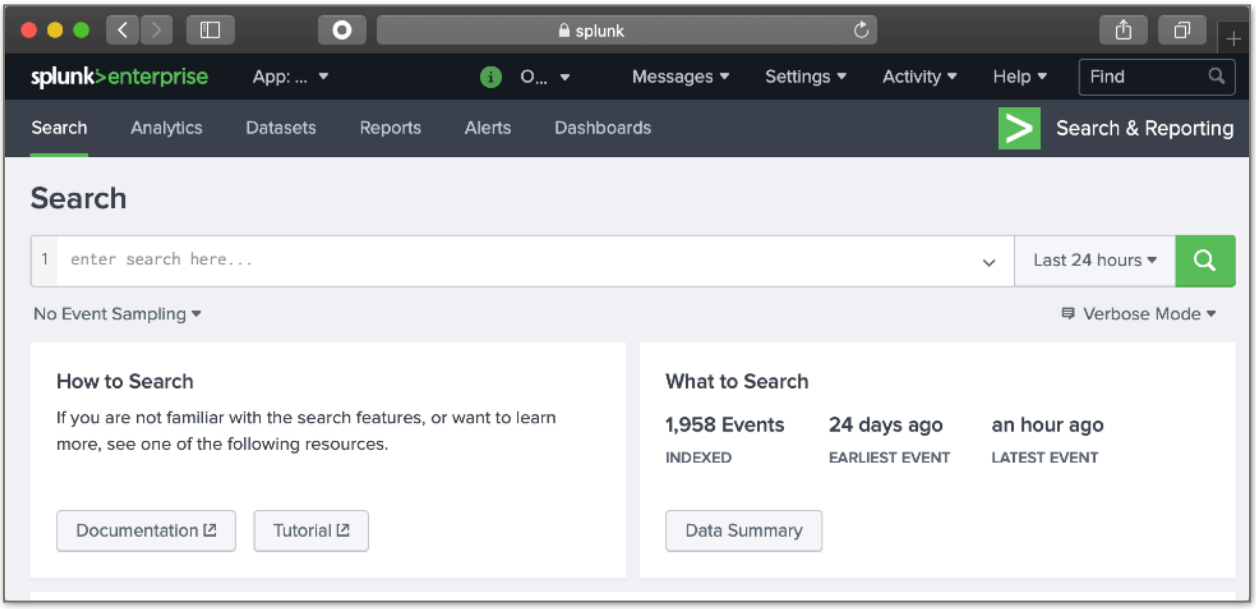
The hosts tab will show the hosts that are retrieving data and those that are submitting it. If you entered a host name override when setting up the Jamf Pro listeners, you'll see that name listed. If you left it blank, you see the Jamf data coming from the Splunk server since that's what's retrieving the API data. Switch to the Sources tab to see each of the individual inputs and you'll be able to watch the record counts grow as the Add-on completes its first round of data retrieval.



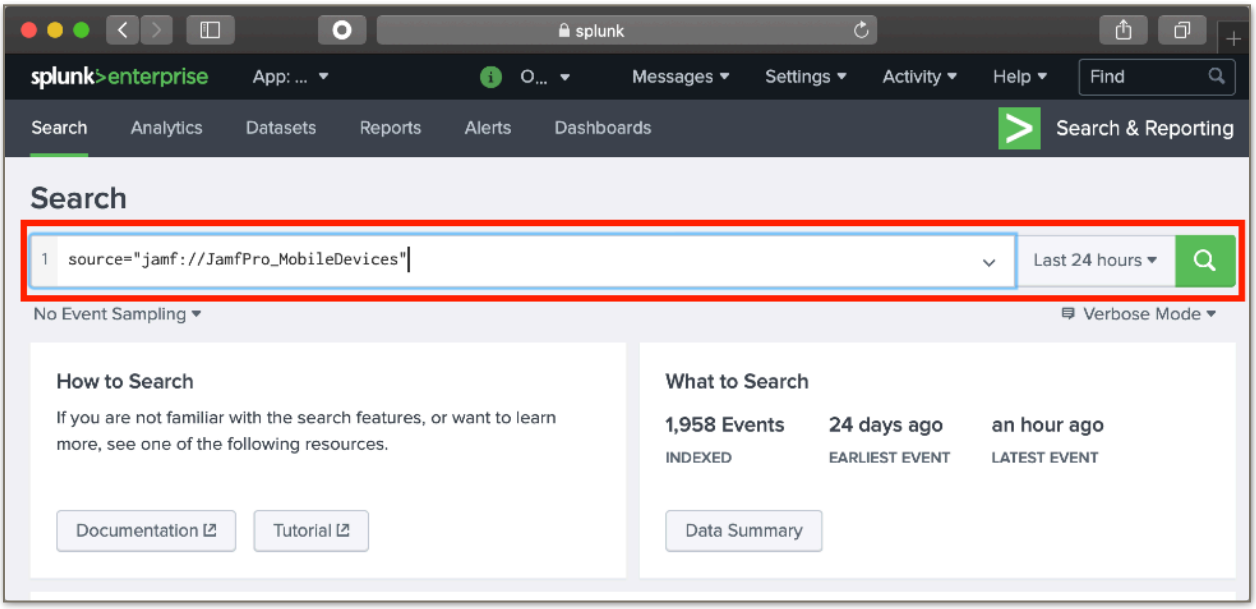
Analyzing Jamf Pro Data with Splunk

Viewing event data in Splunk's Search Screen

Now that some events are starting to populate on our listeners, we can start using it. Start from Spunk's home page, click into the "Search and Reporting" app. We'll start in the app's "Search" tab.

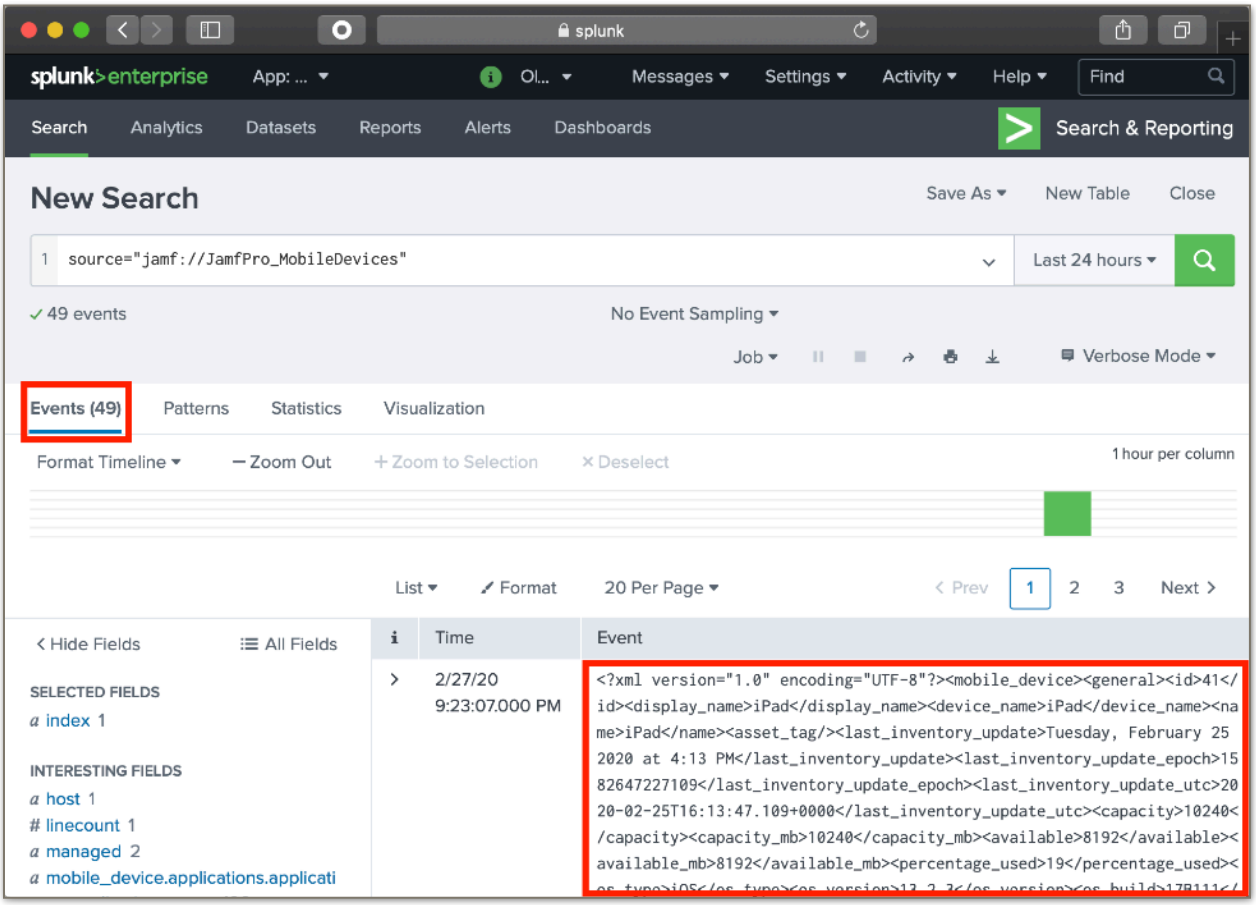


The search page is where we extract useful data and statistics from the raw data that's been imported and indexed in Splunk. Much of Splunk's functionality rests on its search language. We'll be learning to use that language in this section. In the example shown here, we've created an input for the mobile devices end-point with "JamfPro_MobileDevices" as the source Name. Enter `source="jamf://JamfPro_MobileDevices"` into the search box. The time interval is set to "Last 24 hours". Click the green search button or press enter.



Splunk will search its database for event records that match the search command and display them. Observe that the Events count in this example is "49". This is because we have 49 mobile devices in our database that were imported in the last 24 hours, the search interval we selected.

On the lower- right, you can scroll through a display of the raw data for each event -- this is the XML that was retrieved from the Jamf Pro API when the Add-on application retrieved them.



Let's take a look at our search command so far: `source="jamf://JamfPro_MobileDevices"`. This is an example of a data filtering command. These commands often appear at the beginning of a Splunk search program. They're what we use to separate the data we want to report on from all the other data contained in Splunk. Let's break that down...

Search Component	Description
`source=""`	This is the field we're filtering on, followed by an = sign. The value we're looking in this example is a double-quoted string literal.
`jamf://`	This indicates that the data came in via the jamf Add-on.
`JamfPro_MobileDevices`	The source name we entered when we configured the listener for this data.

Search Concepts

So now we have a selection of event records. It's good to see the XML data is there, but we need to extract individual fields to produce any kind of usable report. To do so, we're going to add to our search program. The examples in this section will use the data provided by Jamf Pro's "**`/JSSResource/computers`**" end point. Let's go over some basics before we start looking at building an actual report.

The Basics, #1: Limiting Results to the most recent information

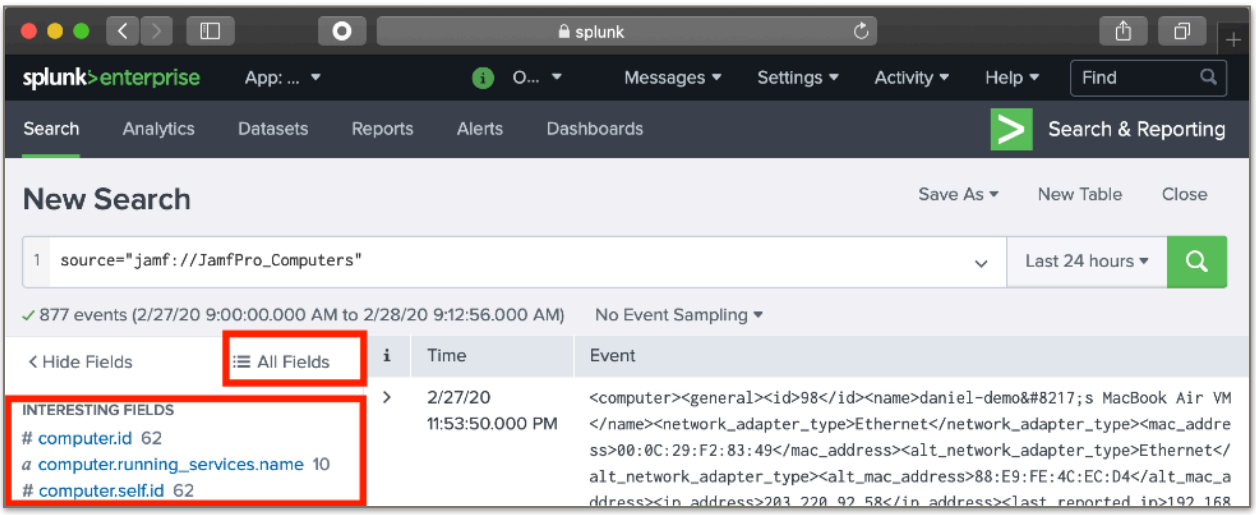
So far, our search consists of `source="jamf://JamfPro_Computers"`. Since we just started importing data, we only have one set of events for each computer. Over time, many reports will accumulate for each computer and `source="jamf://JamfPro_Computers"` will return all of them. That's not what we want -- unless we need to report on changes over time to observe historical trends, reports will only want to observe the current state of the devices. We can limit the events returned by our search by adding a "dedup" command...

```
source="jamf://JamfPro_Computers"
| dedup computer.general.id
```

Note the | (pipe) character at the start of line 2. This is just like in most shell languages -- we're indicating that the results of the first line will be pipelined into the second. That's followed by a dedup command. This command tells Splunk to remove all but the most recent data for each computer based on the value of the `computer.general.id` field, Jamf Pro's primary key for each computer. It does this using the internal timestamp that Splunk adds to every event it imports. After adding the dedup, the result set will consist of only of the most recent data for each computer.

The Basics, #2: Extracted Fields

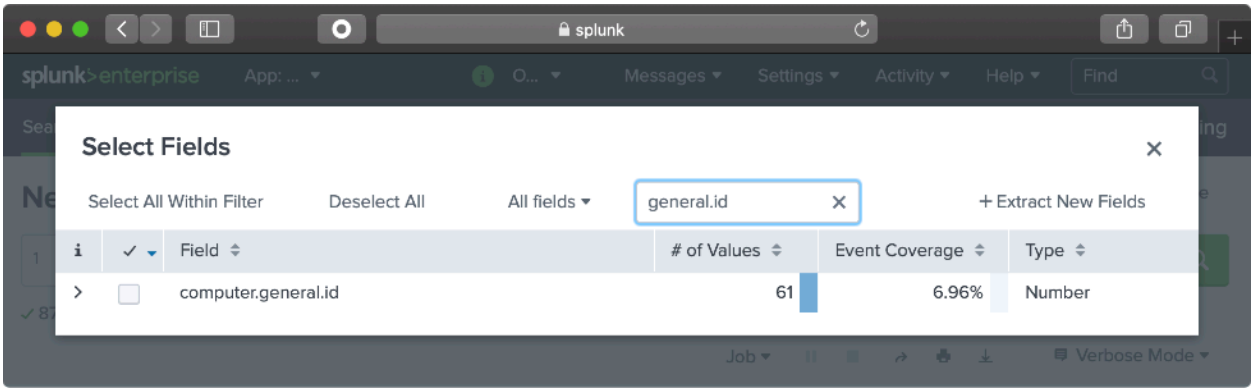
The raw XML isn't useful for reporting as-is, but Splunk has already extracted and indexed each XML data value as it was imported. The fields contained in the raw XML records are listed along the left hand side.



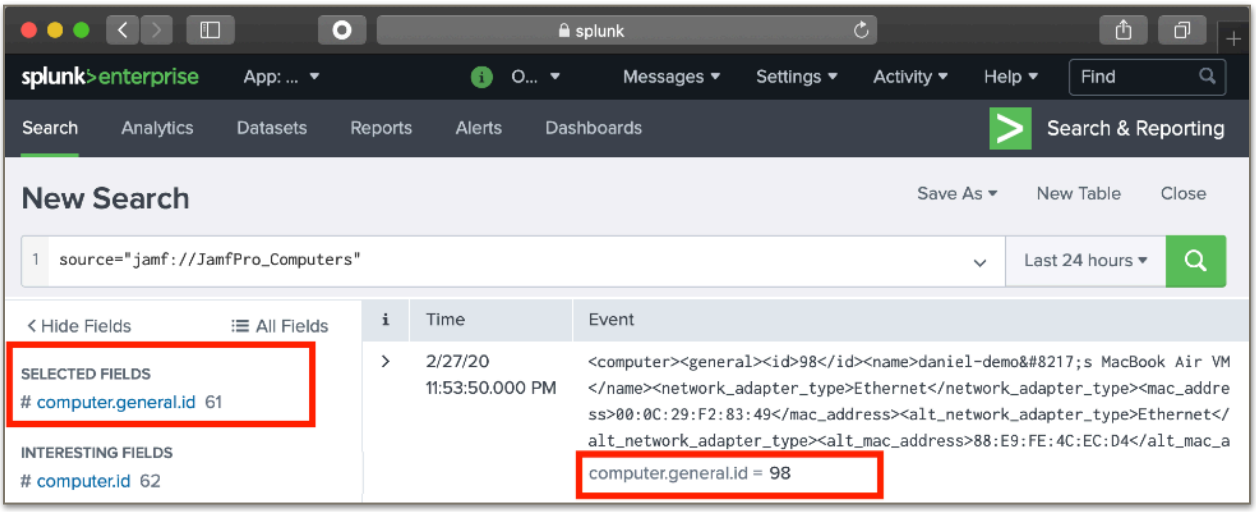
The icon next to each field indicates its type (number, string, etc.) and the number to the right shows the number of unique values found in the current data set for that field. In this example, there are 62 unique computer IDs because our data set contains 62 computers.

The field names represent the values' tag hierarchy in the raw XML, so if we imported and event like "`<computer><general><id>98</id></general></computer>`". Splunk would create a "`computer.general.id`" field and the value would be "98".

You can scroll through the list, but since there are so many, it's much easier to click the "All fields" button and enter part of the xml tag you're looking for in the filter box...



Check the box next to some fields of interest and close the window. The fields you selected are listed under "Selected Fields" and their values will be shown after each row of raw data.



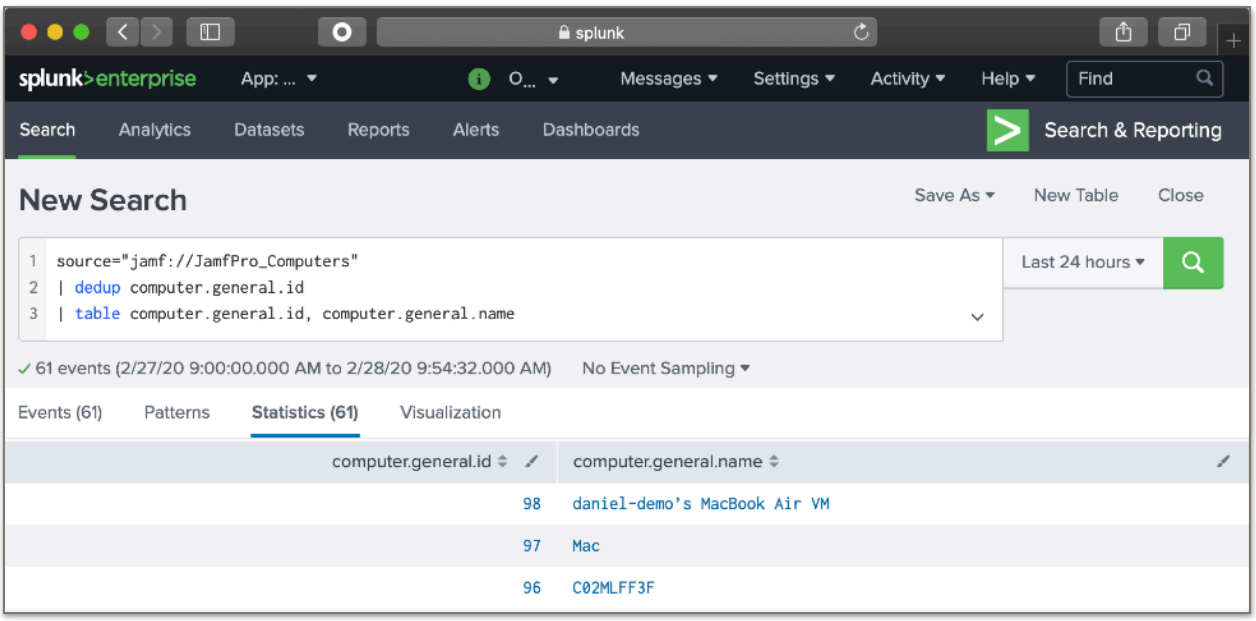
The Basics #3, Our First Complete Search

Let's combine what we've done so far to create a result that we could include in a report. Enter this into the the search box and press enter:

```
source="jamf://JamfPro_Computers"
| dedup computer.general.id
| table computer.general.id, computer.general.name
```

Here, we have added a third line to our search. The table command is a Splunk "Statistics" option. It writes the requested fields to a columnar list we could then include in a report.

Splunk will switch over to the Statistics sub-tab and you'll see the requested table...



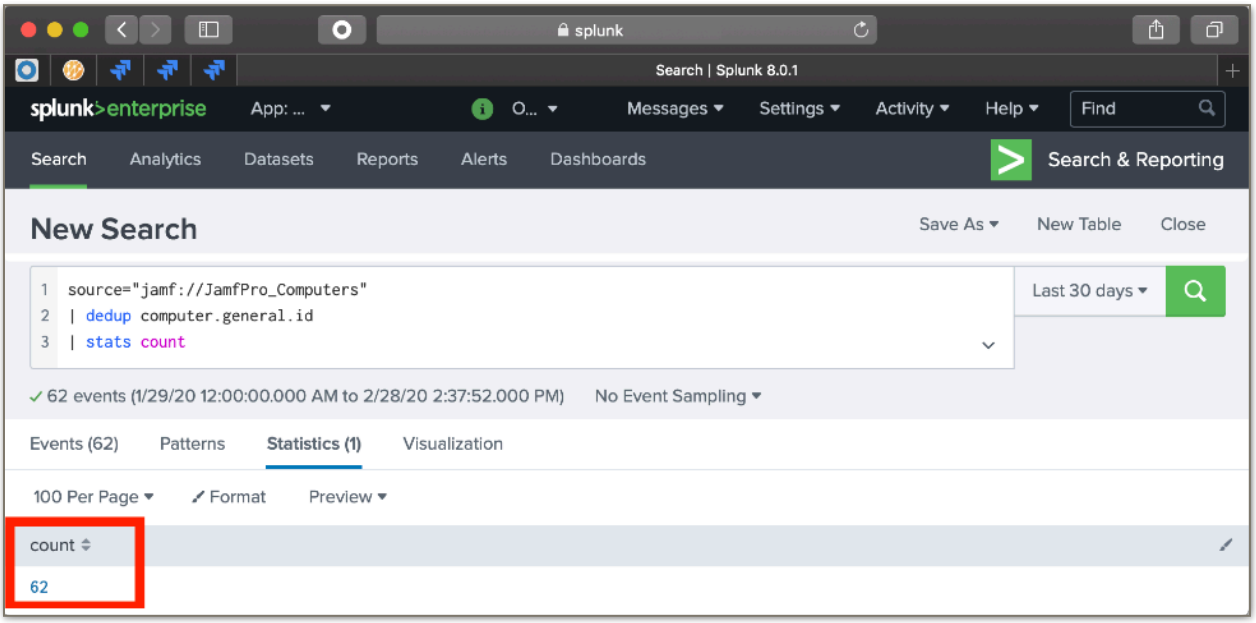
The Basics #4, A Search with Numeric Results

Sometimes there are cases where tabular data could be included in a dashboard. For example, we could include a list of the 10 most used apps by name. But it's more often than not, tabular output is exported as a report, and perhaps then used in a spreadsheet application like Excel or imported into another database. Most dashboards consist mainly of graphs and other visualizations. Graphs can quickly summarize vast amounts of information and data visualizations are a powerful means of observing the relationships and trends hidden within our data. To produce a graph, we need numeric data, such as counting the number of devices running each application.

Let's create a simple search with a numerical result, then add the result to a dashboard. To begin, enter the following search program:

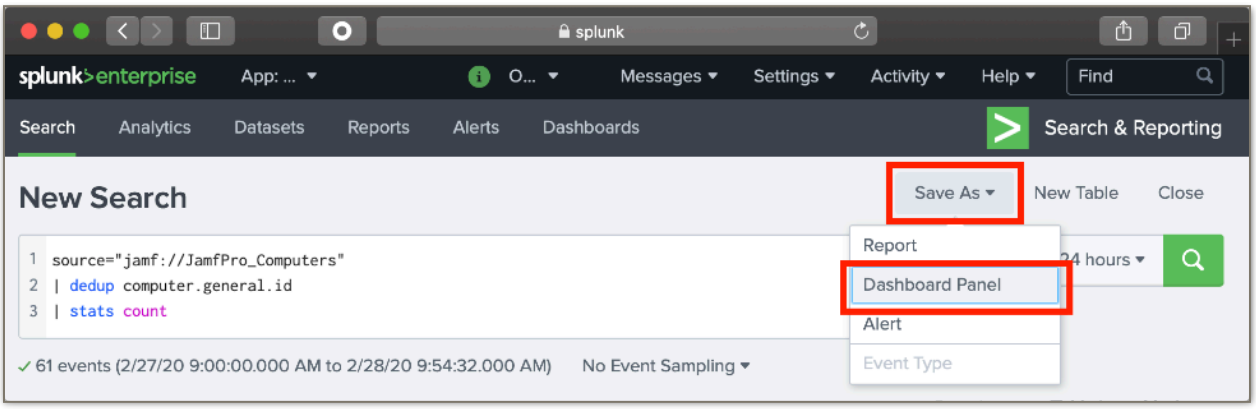
```
source="jamf://JamfPro_Computers"
| dedup computer.general.id
| stats count
```

This search is identical to the one above, but instead of outputting our data set to a table, we use the stats command to simply return the number of records returned by the preceding search commands.

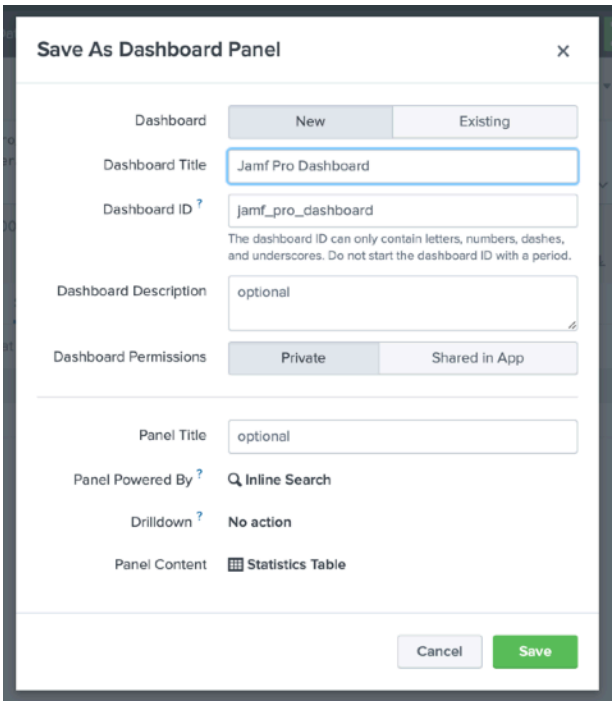


Moving Results to a Dashboard

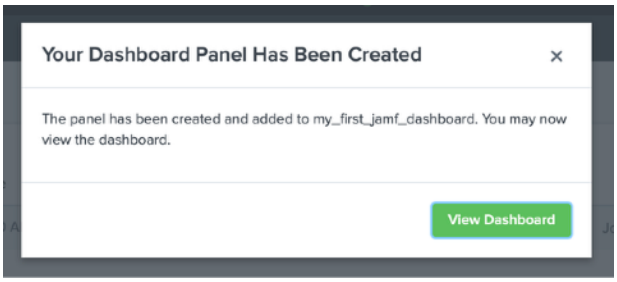
Splunk search results can be saved to Reports, Dashboards, and Alerts. Let's create a dashboard. Click "Save As" and choose "Dashboard Panel".



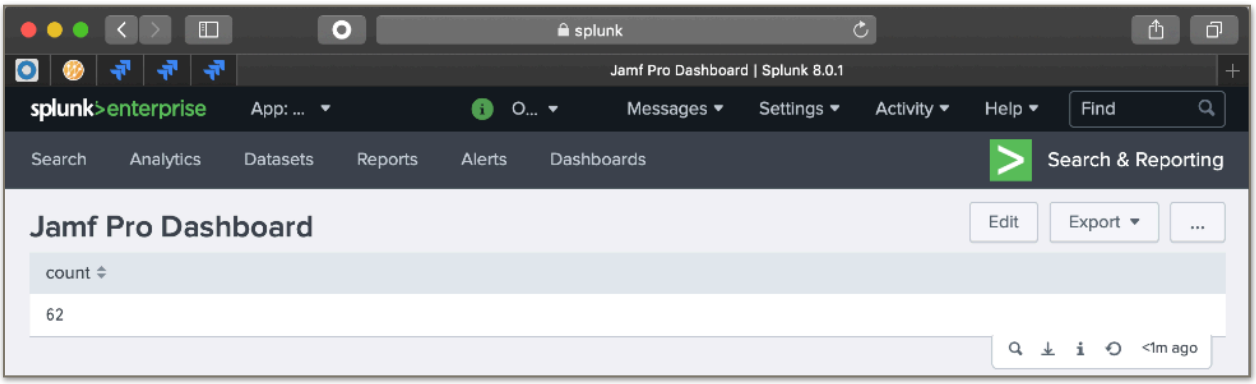
In this case, we are going to create a "new" dashboard. As we create additional panels, we'll usually be adding them into an existing dashboard so we can start combining different panels to tell our story. You can give the new dashboard a Title. We can set this to "Private" for now since we're still working on it. Dashboard permissions can be adjusted later when we're ready to share our work.



Click the Save button. Splunk will tell you that your dashboard has been created.



If you click "View Dashboard", you'll see the search result, but now it's in a dashboard...

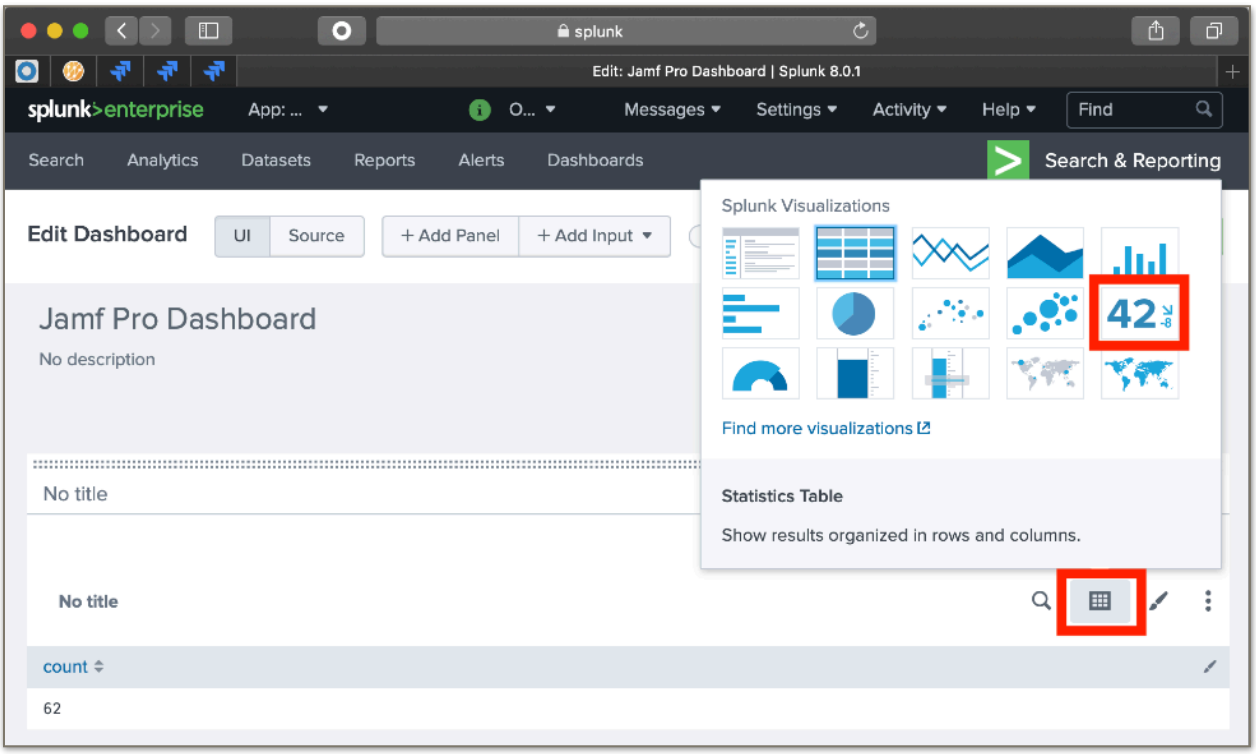


Adjusting the Dashboard Panel's Appearance

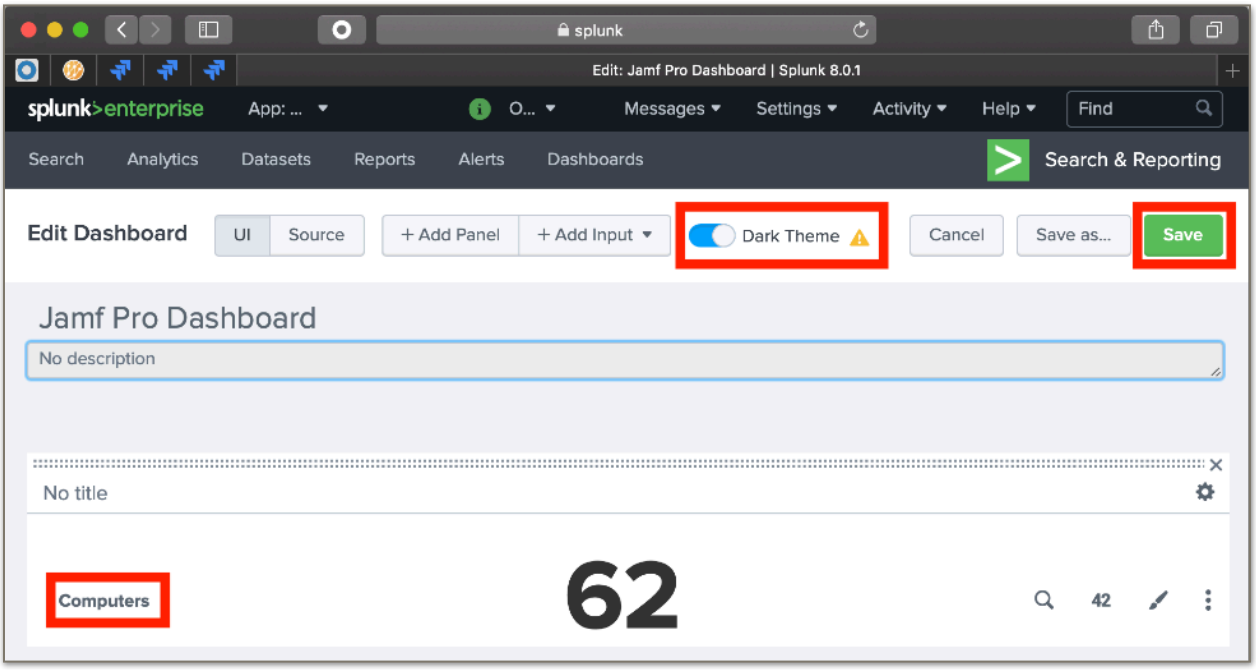
Initially, our search results are simply appended in table form to a dashboard, but that's probably not the presentation we're going for. To convert the table to a visualization, click the "Edit" button at the top-right of the dashboard.

In Edit mode, each dashboard panel will have a button to change the visualization format and one to set additional appearance settings, like how to display labels on the X & Y Axis, whether or not to show data values, if a legend should be displayed, etc.

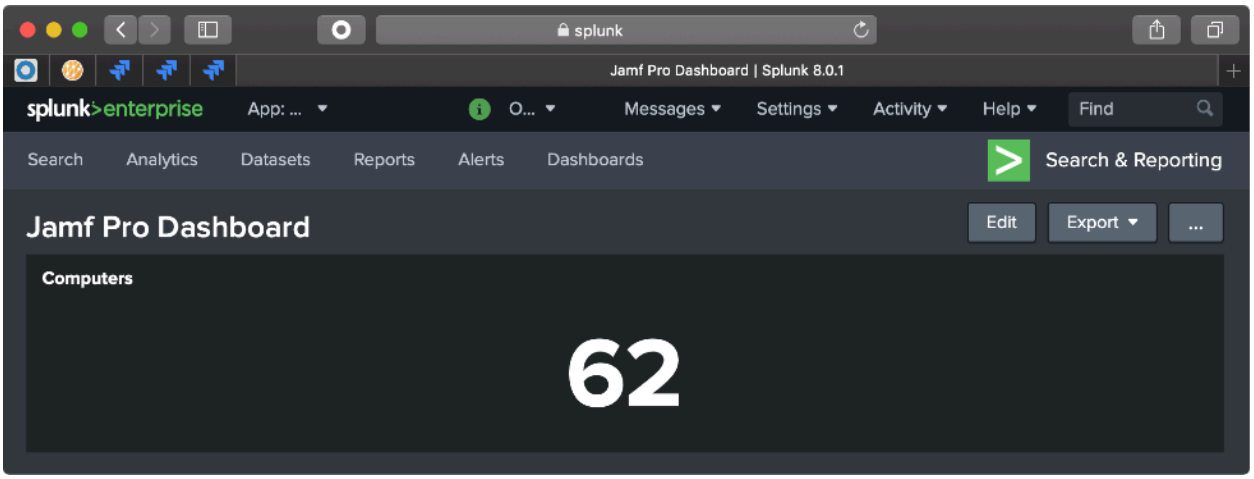
Click the "Single Value" visualization option for our search. This visualization is a good way to prominently display an important metric.



Now you'll see that instead of a table with a single value, the computer count has changed to just a large number. You can enter "Computers" as a heading for the panel. While we're still in edit mode, click the "Dark Theme" slider if you prefer that kind of appearance.

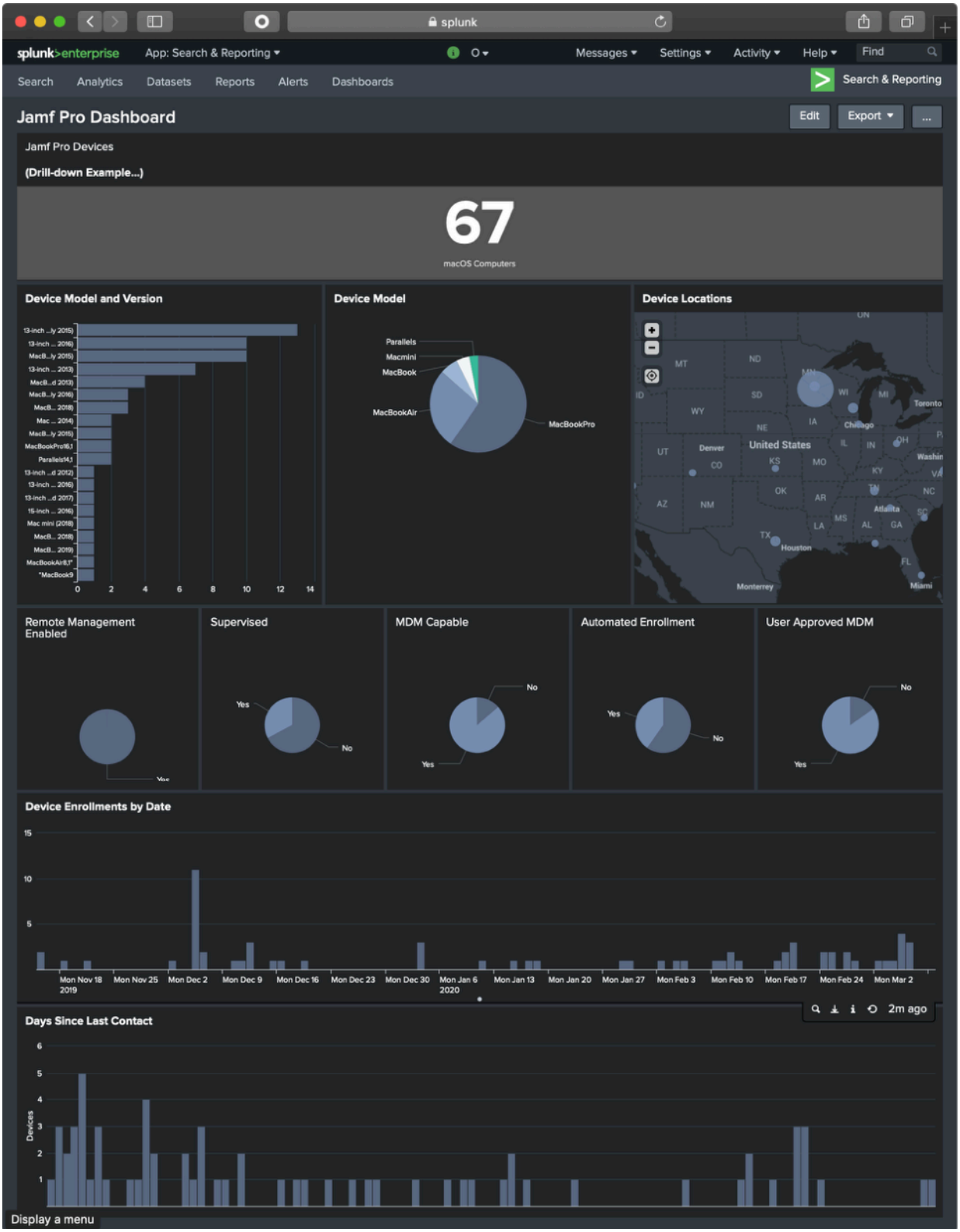


Then click the "Save" button at the upper-right. If you requested dark theme, you'll be prompted to reload the page, and then we'll see our dashboard with its newly-formatted panel...



A Completed Dashboard Example

In the following sections, we'll show some typical search techniques when working with Jamf Pro API data. Some of these were used to build this simple dashboard example:



Search Examples, Illustrated

Here's the simple search we used when we were going through the steps in the previous section. We'll build on what we learned from it in the examples that follow.

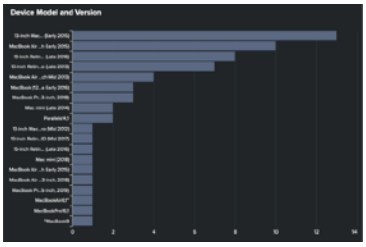
Example	Total computer count	
Issue	**"I have an important metric that I want to display prominently."**	
Steps	1. Use data from the Jamf Pro Computers listener 2. Filter for the most recent data for each computer 3. Count records	

Search	`source="jamf://JamfPro_Computers"`
	`| dedup computer.general.id`
	`| stats count`	

Format	`Single Value`	

Next we'll compare hardware models. What's the most common model? The least? Are there a lot of old models still running out there? A graph makes it easy to assess the proportions of things.

The computer model information is not found in the /General data records we searched in the previous search example. Jamf's computer endpoint can return more data than can fit in Splunk's default setting for the the length of any one data row, so to break things up into smaller chunks, the Add-on splits each computer event into a "general" header event, and a series of child "subpages". The subpage records contain additional details about the computer. One of these children will have the model information. Note that in these "child" detail records, the computer ID is in the field "computer.self.id".

Example	Hardware Models	
Issue	**"I want to compare proportions of a series of values."**	
Steps	1. Use data from the Jamf Pro Computers listener 2. Get the records that have model information. 3. Filter for the most recent data for each computer 4. Get the count for each model 5. Sort so the most common models are listed first	

Search	`source="jamf://JamfPro_Computers"`
	`computer.hardware.model=*`
	`| dedup computer.self.id`
	`| stats count by computer.hardware.model`
	`| sort count desc`	

Format	Bar Chart	

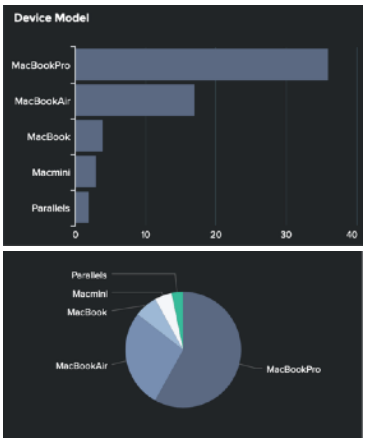
In that example, we reported out each individual model and sorted by count/descending, then displayed as a horizontal bar chart. But you can change all of these things to best suit the story you're trying to tell; the correct search and visualization are determined by the business question you're trying to answer.

You can also group data into categories to get a higher-level view. Let's look at the device model search from another angle: Do we have mostly MacBooks or desktop computers? We'll need to categorize the data by model type (MacBook, iMac, MacMini, etc.) instead of the specific model instance (e.g. "MacBookAir 11-inch Late-2018").

Pie charts are helpful when we are trying to show how different values make up a proportion of a whole. We could have used a pie chart to show the device model counts instead of the bar chart, but a pie chart doesn't work well if we are showing each individual model and version -- there are so many different models in our fleet that there aren't enough colors in a pallet to get a good contrast from one datapoint to another, and there are a some models where the count is so small relative to a dominating value (e.g., an organization might have 1,000 MacBooks but only 2 MacPros...), that the smaller value's slice wouldn't even be perceptible in a pie chart.

Sometimes the difference between small proportions and large is so great that we have to resort to logarithmic scales if we want to visualize them, but non-scientists can't decipher those so easily.

But when we group our computers into hardware model-type categories, the number of data points decreases and their value increases. A pie chart might be good way to show how our categorized numbers contribute to a whole.

Example	Hardware Models, Categorized	
Issue	**"I want to compare the proportions of a grouped series of values."**	
Steps	1. Use data from the Jamf Pro Computers listener 2. Get the records that have model information. 3. Dedup for the most recent data for each computer 4. Copy the model ID to a variable called "model" 5. Use a regular expression to extract the model type 6. Get the count for each model type 7. Sort so the most common model types are listed first	

Search	`source="jamf://JamfPro_Computers"`
	`computer.hardware.model_identifier = *`
	`| dedup computer.self.id`
	`| eval model='computer.hardware.model_identifier'`
	`| rex field=model "(?<modelType>.*?)\d.*"`
	`| stats count by modelType`
	`| sort count desc`	

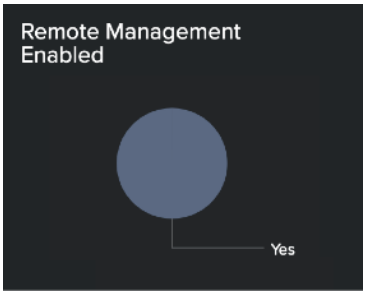
Format	Pie Chart or Bar Chart	

There's a lot going on in that search, but basically we just took the same "`computer.hardware.model_identifier`" model-name field we used in the last example, used a regular expression to extract just the model type, and then did our counts on that.

Pie charts and stacked columns or bars (either as values or percentages) can also be useful for showing the proportions between true/false values. Are our devices encrypted or not? Are they running the latest release of the Operating System? Are they a member of a "compliant devices" smart group? What is the success-to-failure ratio of the Microsoft Office update we started pushing out last night?

If we use a pie chart to show the percentage of devices that are in some "desired state", we'd want one big circle with a single color showing. If we see two colors, we have a quickly-recognized indication that something has gone wrong with our device management strategy.

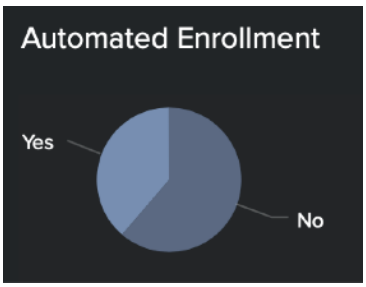
In this example we'll replace "true/false" raw values with "yes/no" since that will look nicer in the chart. This doesn't write anything back to Splunk's database... we're just changing the Search's output.

Example	Remote Management Enabled	
Issue	**"I want to show the relationship between yes/no values but expect zero no's"**	
Steps	1. Use data from the Jamf Pro Computers listener 2. Filter for the most recent data for each computer 3. Replace True/False with Yes/No 4. Get the value counts for a boolean field	

Search	`source="jamf://JamfPro_Computers"`
	`| dedup computer.general.id`
	`| replace "true" with "Yes", "false" with "No" in computer.general.remote_management.managed`
	`| chart count by computer.general.remote_management.managed`	

Format	Pie Chart	

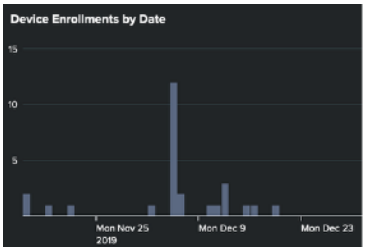
Pie charts can also be used for boolean values where either true or false is an acceptable value, but we're interested in the proportion between the two. For example, suppose we recently added a new enrollment method that uses Apple's Automated Enrollment Program where we used to only support User-Initiated (web) enrollment. We might want to see how many people are taking advantage of the new method. We could make a chart to show that and watch it grow over time as the new method gains popularity.

Example	Use of Automated Enrollment	
Issue	**"I want to show the relationship between yes/no values"**	
Steps	1. Use data from the Jamf Pro Computers listener 2. Filter for the most recent data for each computer 3. Replace True/False with Yes/No 4. Get the value counts for a boolean field	

Search	`source="jamf://JamfPro_Computers"`
	`| dedup computer.general.id`
	`| replace "true" with "Yes", "false" with "No" in computer.general.remote_management.managed`
	`| chart count by computer.general.management_status.enrolled_via_dep`	

Format	Pie Chart	

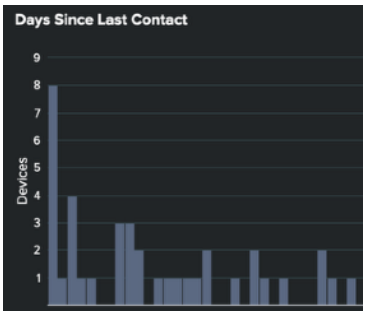
Time-series analysis is a great tool for both spotting anomalies and forecasting. For example, if we graph hits on a web server over time and we see a major spike pop up, either that new marketing campaign is going really well or we are getting hit by a denial-of-service attack. Over time, the consumers of your visualizations will grow accustomed to the normal ebb and flow of things and will be able to spot anomalies when they occur. These kinds of graphs can support daily operations too. For example, it might be helpful to watch new devices appear as the desktop teams are out in the field doing device refresh.

Example	Device Enrollments by Date	
Issue	**Charting Event Instance Count by Date using UTC Timestamp**	
Steps	1. Use data from the Jamf Pro Computers listener 2. Filter for the most recent data for each computer 3. Use a regular expression to extract datetime string from UTC 4. Convert datetime string to a Splunk-format epoch time 5. Export the results as a time series grouped by day	

Search	`source="jamf://JamfPro_Computers"`
	`| dedup computer.general.id`
	`| rex field= "(?<dateTtime>....-..-..T..:..:..).*?"`
	`| eval _time = strptime( 'computer.general.last_enrolled_date_utc', "%FT%T.%3N%z")`
	`| timechart span=1day count by true()`	

Format	Column Chart	

The example above shows how to use RegEx to manipulate dates into a format usable by Splunk, but it's usually a lot easier to just convert Jamf's millisecond-based offset epoch times to Splunk's second-based epoch format. We'll do it that way in this example where we'll look at how many days have elapsed since our devices last checked in with Jamf Pro. Ideally, we'll see a very tall first column.

Example	Days Since Last Device Contact with the Management Server	
Issue	**Counting events using epoch-time formatted data**	
Steps	1. Use data from the Jamf Pro Computers listener 2. Filter for the most recent data for each computer 3. Convert epoch time format and get offset since now. 4. Count the events by day, fill in any missing days as 0's 5. Rename the fields with labels we want in the graph	

Search	`source="jamf://JamfPro_Computers"`
	`| dedup computer.general.id`
	`| eval daysSinceLastContact=floor((now()-('computer.general.last_contact_time_epoch'/1000))/(60*60*24))`
	`| stats count by daysSinceLastContact`
	`| sort daysSinceLastContact`
	`| makecontinuous daysSinceLastContact`
	`| rename count as "Devices", daysSinceLastContact as "Days Since Last Contact"`	

Format	Column Chart	

An end-to-end Ad Hoc Reporting Example

Suppose we're preparing for an audit and want to get a report of all Macs that have an unapproved administrator account on the machine. As we build this report, we'll demonstrate how to find the Splunk field that contains the data we need and explore a more advanced concept -- how to merge data from separate records into a single report. This example will require us to deal with the fact that Jamf Pro computer details are split between a "General" record and a series of sub-pages with lots more detail, but the data we join could just as easily be from completely different source systems. For example, Splunk has an add-on to read Active Directory data, so we could exclude any users or computers that are members of an approved security exceptions group.

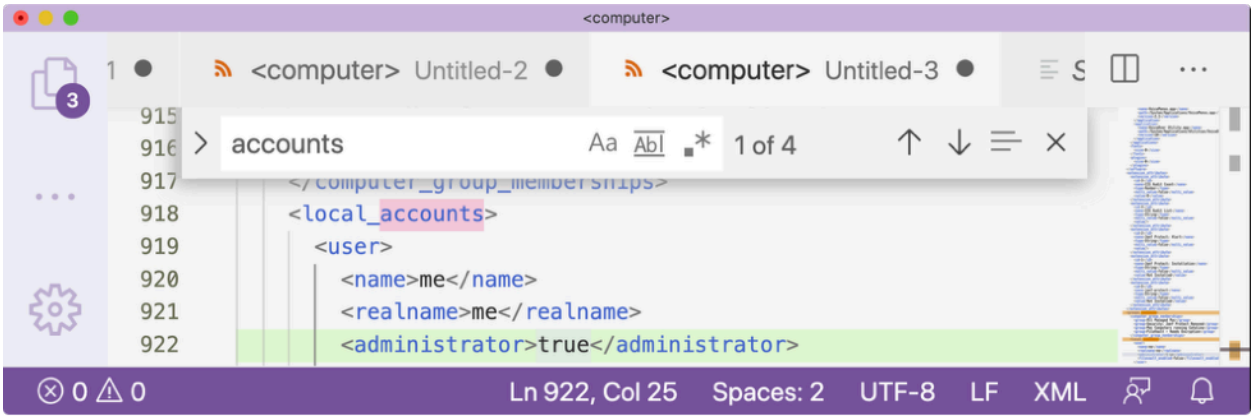
There are lots of powerful applications for this technique. We could join Jamf Pro's installed applications data to Jamf Protect's anomalous event detection data because both feeds contain the device serial number. Then you could join those results to data from server access logs based on IP address, host name, user, or MAC address, and so on. The result would provide fast information about a situation that might have taken days to compile in the past. That's time we don't have in a modern IT operation.

Finding Fields

The first thing we need to figure out is how to get a list of admin accounts on the machines. It seems reasonable that accounts would be computer-related data -- we can see account information when we look at a computer's details in the Jamf Pro application.

But what's the actual field name? We won't have memorized every field name in Jamf Pro -- there are hundreds of them. But you don't have to know a lot about Jamf Pro to guess that the field name will probably contain the word "account" somewhere in its name.

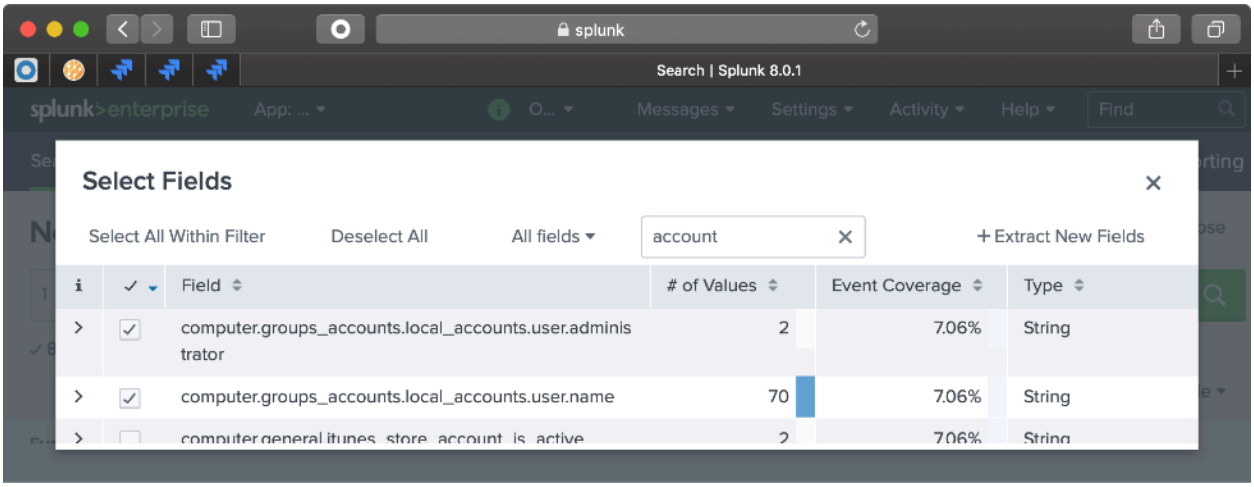
One strategy for locating field names is to pull up a computer record in https://<your instance name>.jamfcloud.com/api via the /JSSresource/computers/id endpoint. If we do a "find" for the word "accounts" in some sample output, we'll quickly find it's positioning within the XML hierarchy...



Looking at the raw XML is a good way to get a sense of the field structure. We'll be able to see that the feed contains local account user objects and that each of these has a name and a flag to indicate that it is (or is not) an administrator. Administrator in this sample record has a value of "true", and it stands to reason that it would be a boolean. If you look through the XML, we can see that its xpath is `/computer/`

`groups_accounts/local_accounts/user/administrator`. Just replace the slashes with periods and you've got your Splunk field names.

Once you've got a sense of the data structure, you can easily find field names in Splunk without having to pick apart pages of XML. If you know the API endpoint and can guess part of the name, just load the source data in Splunk with a search like `source="jamf://JamfPro_Computers"` then click the "All Fields" button. Enter something in the filter value box to see the fields that are likely candidates...



Create the Search

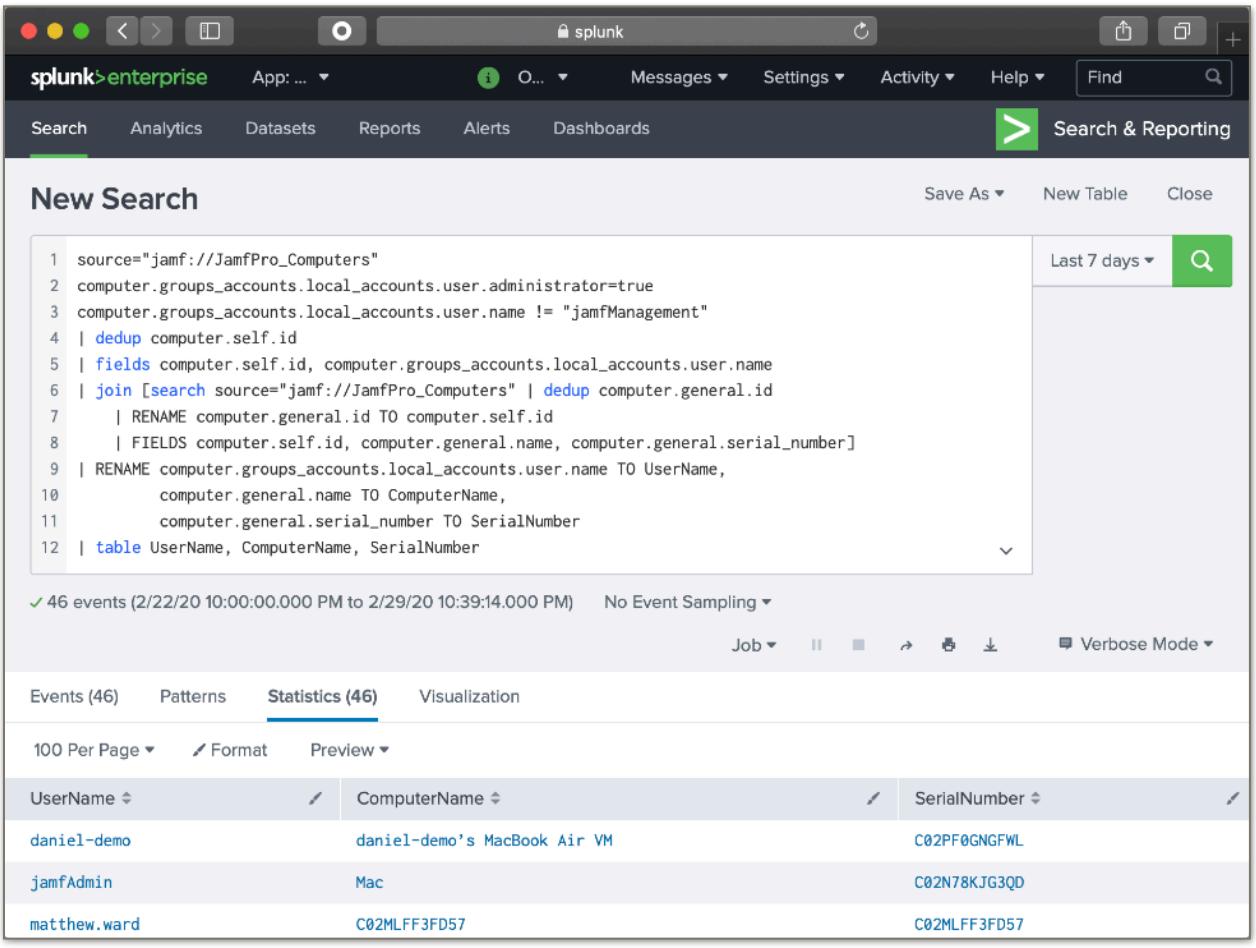
What if we want a list of all of the admin accounts found on our computers. That wouldn't be too difficult to write -- there are computer subpage records that contain that data. But what if we also want to include something from the computer's "general" record, like computer name? Just like in the SQL database language, Splunk allows us to join fields from related records into a single report.

Example	Create a report with data from separate records
Issue	**"I want to find out who's got an unauthorized admin account on their computer."**
Steps	1. Use data from the Jamf Pro Computers listener 2. Find records where the administrator flag is true 3. Dedup on computer ID to get the most recent data for each computer 4. Pipe out the resulting computer iD and admin account user names 5. Do a join to link in data from each computer's General record. Convert the computer ID field in General to match the name of that data point in the subpage account records so they'll link up. 6. Rename some long field names so the report will have nicer column headings 7. Output the merged results

Search	source="jamf://JamfPro_Computers"
	computer.groups_accounts.local_accounts.user.administrator=true
	| dedup computer.self.id
	| fields computer.self.id,
	computer.groups_accounts.local_accounts.user.name
	| join [search source="jamf://JamfPro_Computers" | dedup
	computer.general.id
	    | RENAME computer.general.id TO computer.self.id
	    | FIELDS computer.self.id, computer.general.name,
	computer.general.serial_number]
	| RENAME computer.groups_accounts.local_accounts.user.name TO UserName,
	         computer.general.name TO ComputerName,
	         computer.general.serial_number TO SerialNumber
	| table UserName, ComputerName, SerialNumber

Format	Report

Here's what the search looks like in Splunk, and the results...



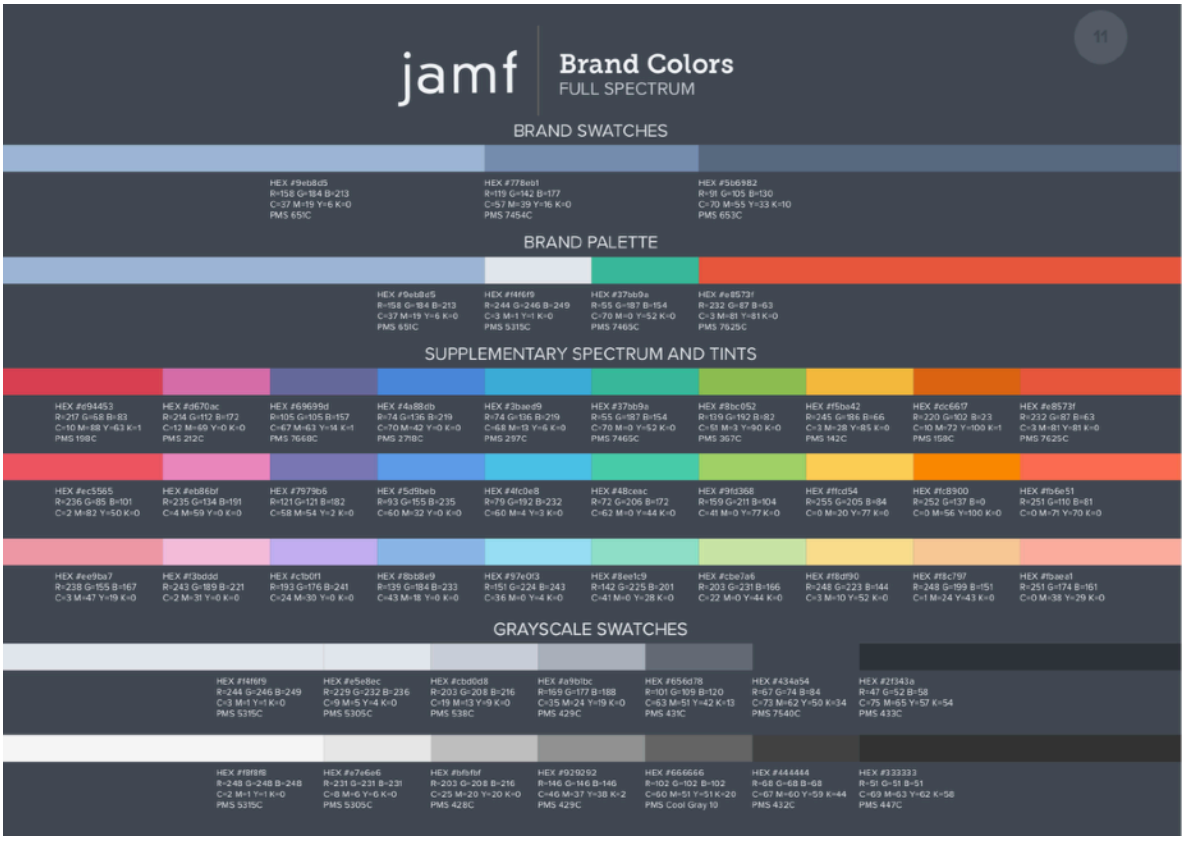
Hopefully we won't see any more cases of unexpected admin accounts this after we remediate and do some (re-)education of our users, but if we want to have on-going monitoring, we could add a "stats" line to our search and place the resulting count on our security dashboard. Then we can use the "Save As" button to create a scheduled report that gets emailed to us every month, or create an alert so that a

security incident is created in the help-desk ticketing system to trigger a response workflow. The point here is that data can power tools that can automate solutions to our device management challenges, even those that involve hard-to-manage user behaviors.

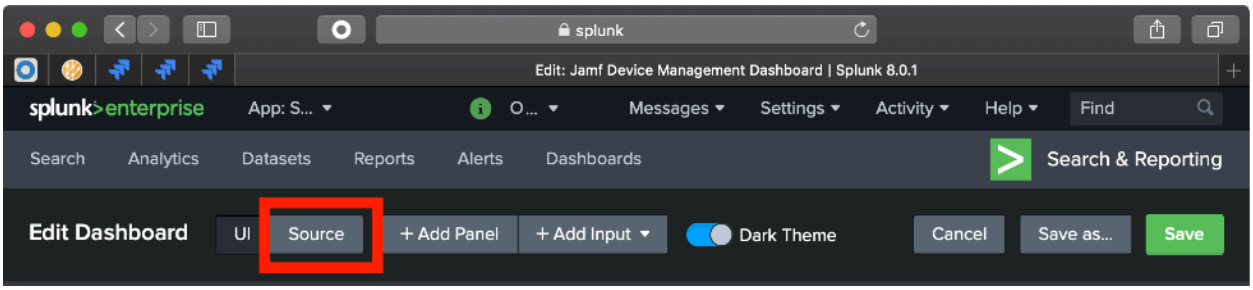
Refining your Dashboard and Adding <options>

Different dashboards and reports can be created for different purposes and users. Once you have your searches written and saved as panels, you can organize and further refine your dashboard by clicking the "Edit" button. The panels can be dragged into new positions to place them into logical order. You can change titles and subtitles. While in edit mode, each panel will have a button to change the visualization type and another to adjust formatting. The options presented will change based on the kind of graph you're showing. For example, for bar and column charts, we can tell Slunk to omit axis labels and legend in cases where the panel's title makes them obvious. This may produce a cleaner appearance.

Splunk makes some default color pallet choices for you when you create a chart. Sometimes these look fine, but if you're creating a dashboard you'll be sharing with others, you'll want to choose your own consistent color scheme. Below is an example of a "Brand colors" pallet. If your organization has a style guide, the brand color scheme has probably been designed to work well for web applications and produce good results when the colors are used together. The resulting look and feel will be familiar to your users, and that makes your dashboards and reports more accessible. Be thoughtful of those who may have visual impairments. You can create versions of your dashboard with color choices that produce a good contrast relative to each other and against your background color.



To change chart colors while a dashboard is in Edit mode, switch from UI to to "Source" mode...



You'll see an editable XML definition of the dashboard contents. Here, you can make many changes that aren't available in GUI edit mode. To change colors, we add a "charting.seriesColors" option. For each panel, find the "charting.chart" option entry, e.g.,

```
<option name="charting.chart">pie</option>
```

...and add a line to add a color option. For a pie chart with only two slices (e.g. yes/no data) we only need to supply two colors:

```
<option name="charting.seriesColors">[0x5b6982,0x778eb1]
```

When formatting a chart with a larger number of data series, you'd supply more colors. This is an example using nine colors from Jamf's brand color scheme:

```
<option
name="charting.seriesColors">[0x5b6982,0x778eb1,0x9eb8d5,0xf4f6f9,0x37bb9a,
0xe8573f,0xd94453,0xd670ac,0x69699d]</option>
```

Many formatting options are available. These are described in the Splunk documentation.

Sending Jamf Protect Data to Splunk

Jamf Protect "Actions" are configured in its web console These determine what the Jamf Protect Agents running on protected devices will do with the log and alert data they collect. The action settings include an option to forward the data to an HTTPS endpoint such as would be needed when using Splunk as a security information and event management ("SIEM") solution.

Configuring Splunk to receive Jamf Protect Events

As a first step, we'll need to configure a Splunk HTTPS Event Collector ("HEC") to receive inbound event notifications from Jamf Protect. Once the setup is completed, Splunk will give us the authentication token we'll need to set up in Jamf Protect so the device agents can open authenticated connections to Splunk. It's easiest if you configure a separate Splunk collector entry for each application that will send data, so even if you've already set up a collector for Jamf Pro webhooks, create a separate collector entry and obtain a new authentication token for Jamf Protect. That way each application's records will have a distinct "source" value in the Splunk database and they'll be easy to isolate when you write your searches.

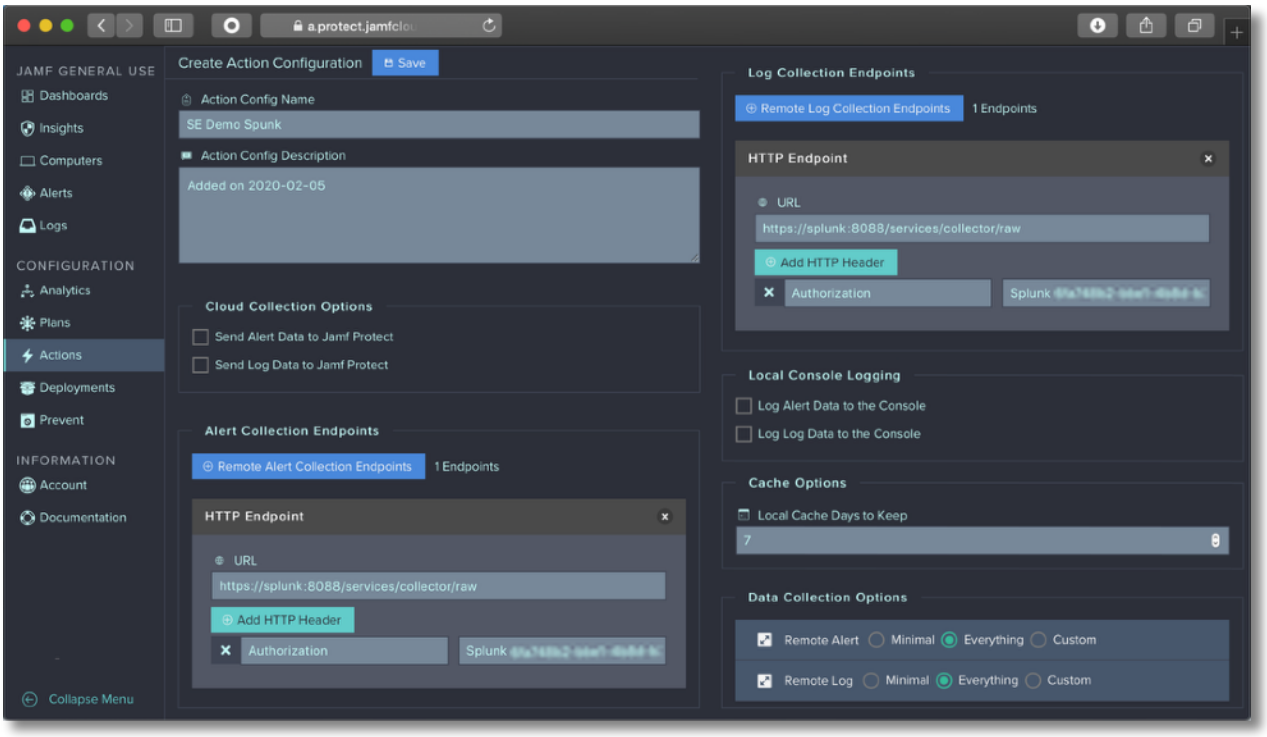
> Instructions for creating an HEC and obtaining an authentication token are available in Appendix 3 of this document, "Configuring Splunk to Receive Event Data".

Configuring Jamf Protect to send events to Splunk

Procedure:

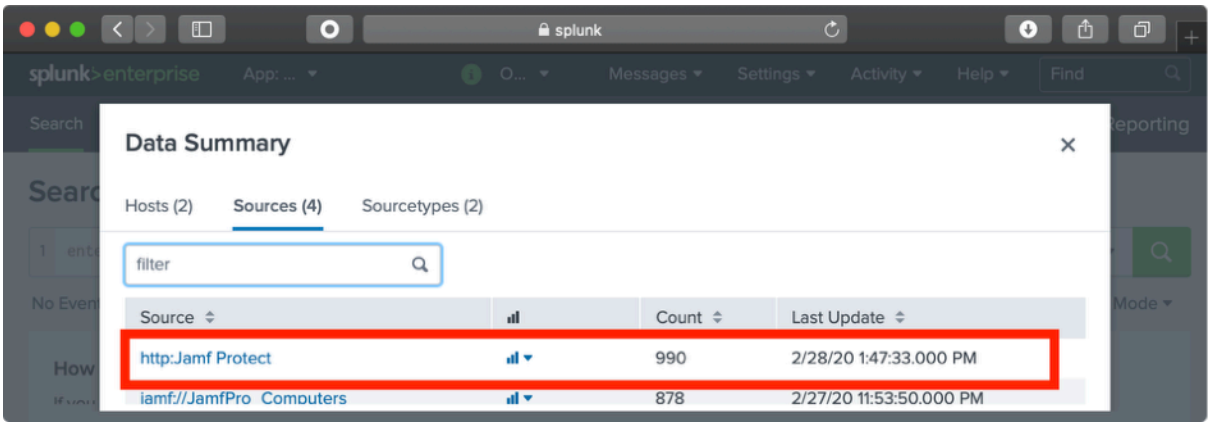
1. Log in to your Jamf Protect instance
2. Click **Actions** from the the left navigation menu.
3. Edit the Action that you've set up and added to your Jamf Protect plan. If you have multiple action/plan combinations in your configuration, you'll need to add Splunk settings to each action that you want to have send data to Splunk. If you haven't yet configured any Actions, please see https://docs.jamf.com/jamf-protect/administrator-guide/Configuring_Actions.html.
4. Create an Action if you don’t have one already, or edit one you’re already using.
    - The URL for the Alert and Log Collectors will be https://yoursplunkhost.your.org:8088/services/collector/raw. The hostname will match your Splunk HEC host (or load balancer if you're using one) and the port will be the one you've enabled in your environment. Default Splunk configurations use port 8088.
    - Set the Header name to "`Authorization`" and the header value to "`Splunk[space character]<The Token you created when creating the connector in Splunk’s Web GUI>`".

This is an example of a Splunk HTTP Event Collector configured in Jamf Protect:

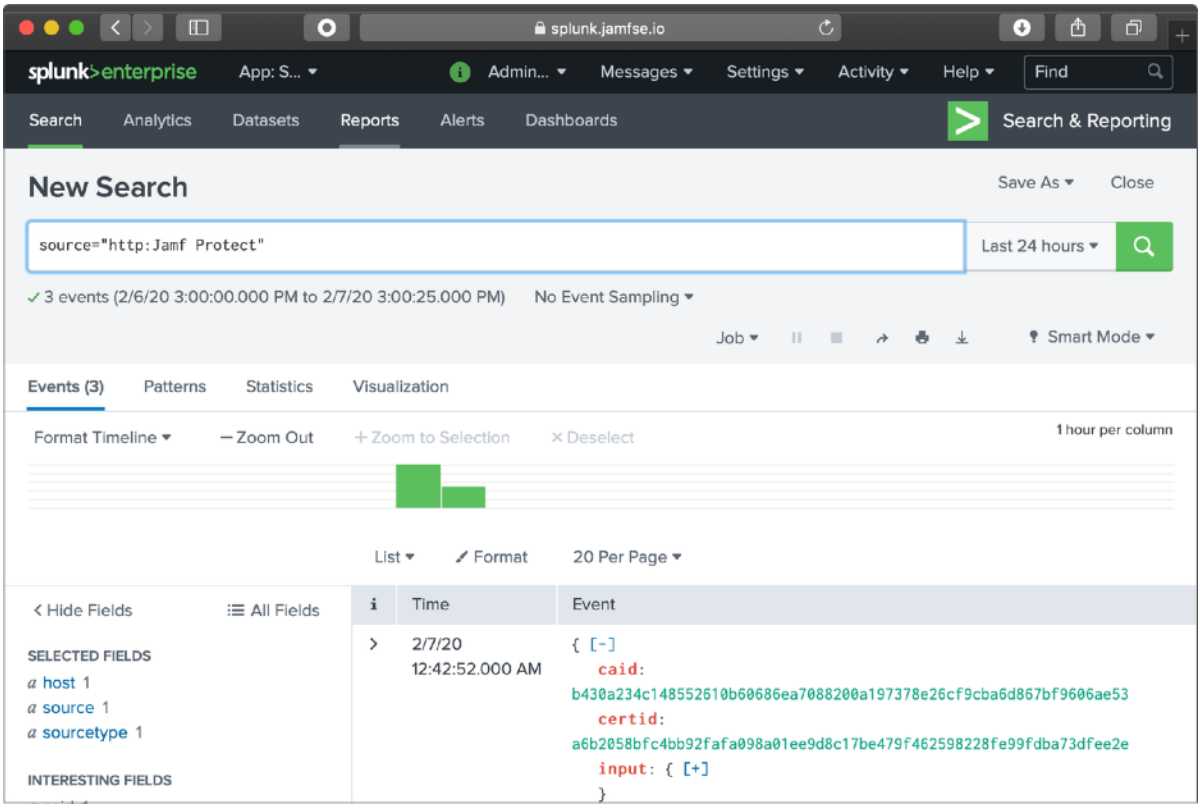


Observe event data flowing from Jamf Protect to Splunk

1. Log back in to your Splunk instance and go into the "Search & Reporting" app.
2. Click the "What to Search > Data Summary" button to see counts of data sent to your Splunk instance. In this example, we entered "Jamf Protect" as the Source Name when creating our HEC. We can see that 990 records have been sent to this source by our Jamf Protect Agents thus far...



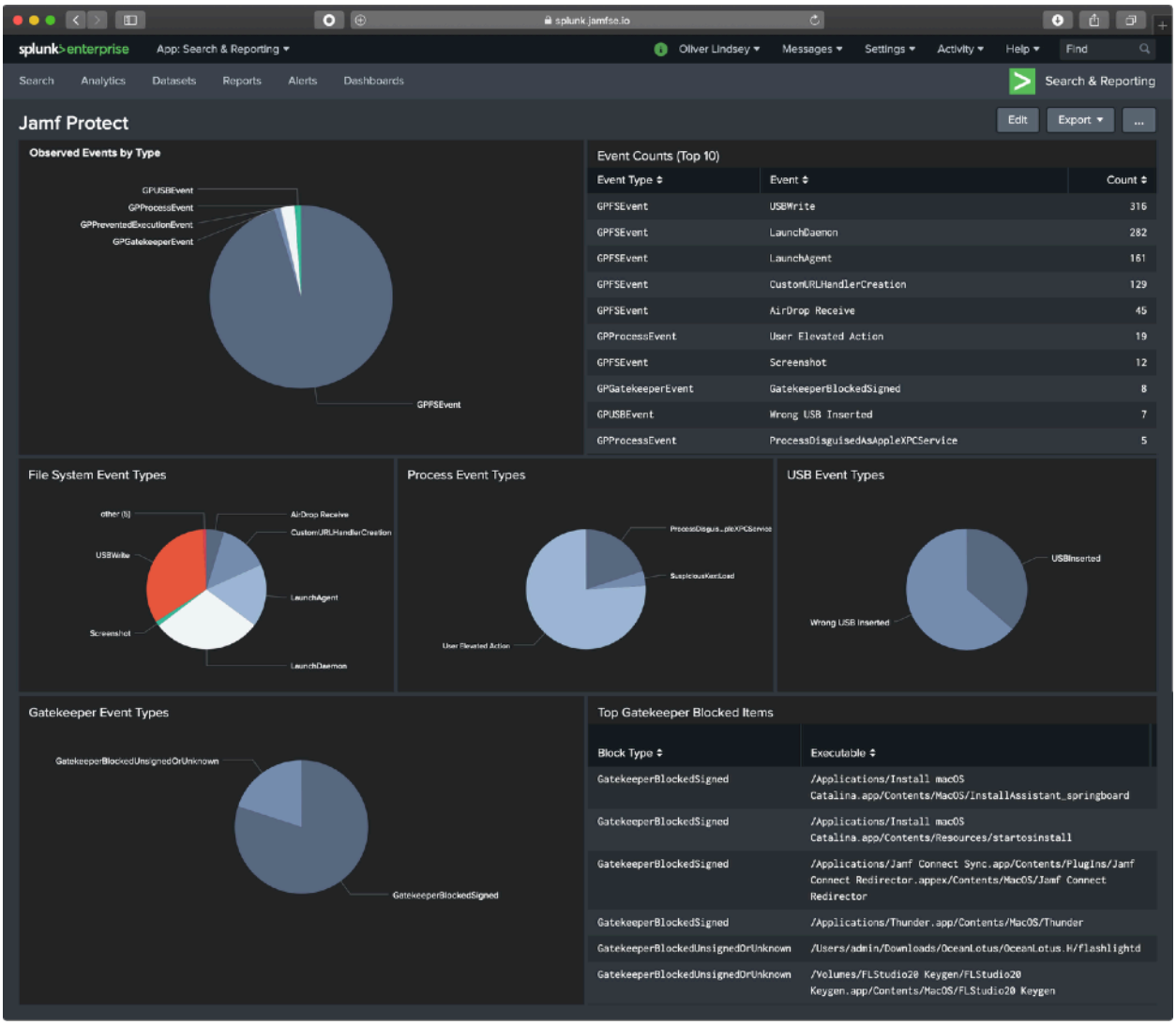
To see event details, do a search on your Protect data source…



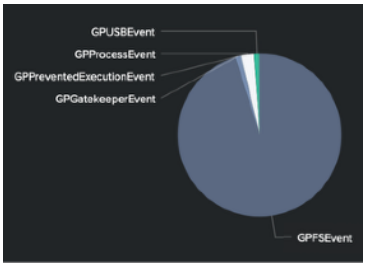
Analyzing Jamf Protect Data with Splunk

The mechanics of searching Splunk data and moving the results to a dashboard are described in the "Search Basics" section of the preceding Jamf Pro discussion so we won't repeat them here. Those basic steps are identical no matter the data source. However, each data source will offer different information, so the questions each can answer is different too. Here we'll focus on Jamf Protect data and its uses, but we'll keep in mind that the most useful insights derived from a data aggregation system often happen when we combine data from multiple systems.

In the following sections, we'll show some typical search techniques when working with Jamf Protect data. Some of these were used to build this simple dashboard example:



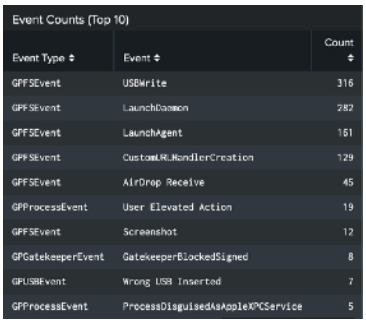
Our first example will show a pie chart of observed security event types. We'd need to drill down further to get useful details, but graphs like this can help us notice if something that is rarely seen suddenly becomes more common. Note that unlike Jamf Pro data where we usually filter data to get the most recent details for a computer (i.e. its "current state"), with Jamf Protect we're dealing with streaming data flowing in real-time. We can set the time range on our searches depending on how granular we want to get. If we are looking to summarize a lot of data, we could look over the last year or month. If we are looking for recent changes, we could time-slice our data into a single day, or perhaps a week if an event happens only occasionally.

Example	Visualizing Event Types	
Issue	**"I want to show the relationship between the count of the different event types"**	
Steps	1. Use data from the Jamf Protect Collector 2. Use stats to count events for each type value in the data 3. Use the rename command to change the field labels	

Search	`source = "http:Jamf Protect"`
	`| stats count by input.eventType`
	`| rename`
	`input.eventType AS "Event Type",`
	`input.match.facts{}.name AS "Event",`
	`count AS "Count"`	

Format	Pie Chart	

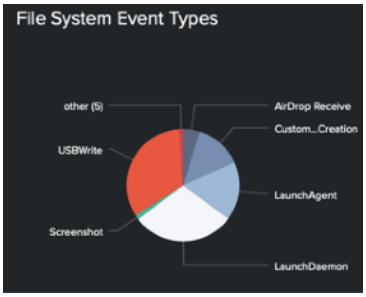
Our next example uses the same basic search as above, but adds-in the names of the actual events. We're starting to dig a little deeper into the details. Often when looking at things like this, we're interested in the most common instances of a thing... "*What apps do our users spend most of their time in?*" "*What kinds of events are expected in the normal course of events and should be logged in case further analysis is needed, but do not otherwise need to be a cause for alarm?*" In other cases, it may be useful to look at the least common instances of a thing. They lead us to notice anomalies... "*All of our machines are running the same apps and OS, how come 5 of them have this new process running?*"

Example	Event Type and Event Name Table	
Issue	**"I want to see the breakdown of the event names by type."**	
Steps	1. Use data from the Jamf Protect Collector 2. Use stats to count events for each type/event pair 3. Use the rename command to change the field labels 4. Sort the results from most common to least 5. Show only the top 10.	

Search	`source = "http:Jamf Protect"`
	`| stats count by input.eventType,`
	`input.match.facts{}.name`
	`| rename`
	`input.eventType AS "Event Type",`
	`input.match.facts{}.name AS "Event",`
	`count AS "Count"`
	`| sort Count desc`
	`| head 10`	

Format	Table	

There seem be a lot of file system events coming in. Let's dig a little deeper into those specifically.

Example	Visualizing events of a single type	
Issue	**"I want to visualize the kinds of File System events we're seeing."**	
Steps	1. Use data from the Jamf Protect Collector 2. Filter for only events of type "GPFSEvent" 3. Use stats to get a count for each event 4. Use the rename command to change the field labels	

Search	`source = "http:Jamf Protect"`
	`input.eventType="GPFSEvent"`
	`| stats count by input.match.facts{}.name`
	`| rename`
	`input.match.facts{}.name AS "Event",`
	`count AS "Count"`	

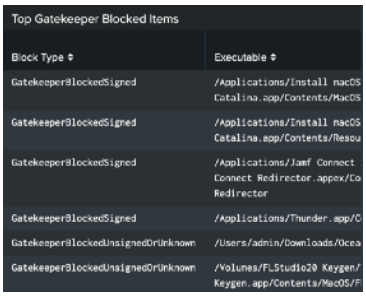
Format	Pie Chart	

We can do the same Pie chart for other event types of interest. The search will be the same, but we will adjust the filter by changing the string after "`input.eventType=`" from "GPFSEvent" to something else. We already have all the strings for the different event types because we listed them out in our tabular event-type breakdown panel -- we use "GPProcessEvent" for Process Event Types, "GPUSBEvent" for USB Event Types, and "GPGatekeeperEvent" for Gatekeeper Event Types.

In the next dashboard panel example, we'll look at how to extract additional details from an event. For example, Gatekeeper events include the path to the executable that was blocked. We might want to take a look at those and figure out where they came from.

Sometimes GateKeeper blocks happen because a user downloaded something completely acceptable but they decided to cancel out of the GateKeeper warning dialog because they didn't want to install or use it right away. Maybe they'll come back to it later. Sometimes it was because they downloaded something they shouldn't have and thought better of installing it when they saw the GateKeeper warning. We're probably more interested in the latter case, but how can we tell the difference?

Over time, we can crowd-source some intelligence about this... if the count on the acceptance of a particular executable across all users is very high, the user hive-mind is collectively telling us it's probably OK and commonly used. If there are very few instances of an executable or a fair number of people are telling Gatekeeper to block it, we may have cause for concern and reason to investigate further.

Example	Reporting on event details	
Issue	**"I want to report on the executables that are being blocked by Gatekeeper."**	
Steps	1. Use data from the Jamf Protect Collector 2. Filter for only events of type "GPFSEvent" 3. Use stats to get a count for each event 4. Use the rename command to change the field labels 5. Use the head command to show the top 10	
Search	`source = "http:Jamf Protect" input.eventType="GPGatekeeperEvent" | stats count by input.match.facts{}.name, input.match.event.path | rename input.match.facts{}.name AS "Block Type", input.match.event.path AS "Executable" | head 10`	
Format	Table or pie chart... or a stacked column or line graph for time-series analysis.	

Sending Jamf Pro Webhooks to Splunk

Summary

How you obtain data from Jamf Pro depends on what data you need and how frequently you need it. Data for an asset accounting report might only be needed once a year, but you'd probably want to know immediately when the Jamf Pro server has been shut down or a device violates an important security requirement.

But in cases where you need to detect a change ASAP, it wouldn't be efficient to set your Jamf Pro Add-on for Splunk input entry to run every second. That would place a lot of API call overhead on your Jamf Pro server and there'd be huge amounts of redundant data in your Splunk database. Over time, both systems would probably slow to a crawl. Jamf Pro's "webhooks" feature will come to our rescue here. Webhooks are an event-driven, outbound HTTP/S POST message sent to any URL you specify when configuring a webhook in Jamf Pro's web GUI. The webhook message includes JSON data with details of the event that triggered the webhook so an integration or data collection receiver gets near-instant notice that something of interest has happened.

The following events are available:

- ComputerAdded
- ComputerCheckIn
- ComputerInventoryCompleted
- ComputerPatchPolicyCompleted
- ComputerPolicyFinished
- ComputerPushCapabilityChanged
- MobileDeviceCheckIn
- MobileDeviceCommandCompleted
- MobileDeviceEnrolled
- MobileDevicePushSent
- MobileDeviceUnEnrolled
- DeviceAddedToDEP
- JSSShutdown
- JSSStartup
- PatchSoftwareTitleUpdated
- PushSent
- RestAPIOperation
- SmartGroupComputerMembershipChange
- SmartGroupMobileDeviceMembershipChange

Details on Jamf Pro Webhook events and the data transmitted is listed in an appendix to this document. More complete and up-to-date information on Jamf Pro webhooks may be found in Jamf's Developer's Portal, https://developer.jamf.com/webhooks

Configuring Splunk to receive Webhooks

As a first step, we'll need to configure a Splunk HTTPS Event Collector ("HEC") to receive inbound event notifications from Jamf Pro. Instructions for creating an HEC and obtaining an authentication token are available in Appendix 3 of this document, "Configuring Splunk to Receive Event Data". Once the setup is completed, Splunk will give us the authentication token we'll need to set up in Jamf Pro so it can open authenticated connections to Splunk. If you have multiple systems feeding events to your Splunk HEC, create a separate application entry for each one so it's easy to filter them by source name when you write your searches.

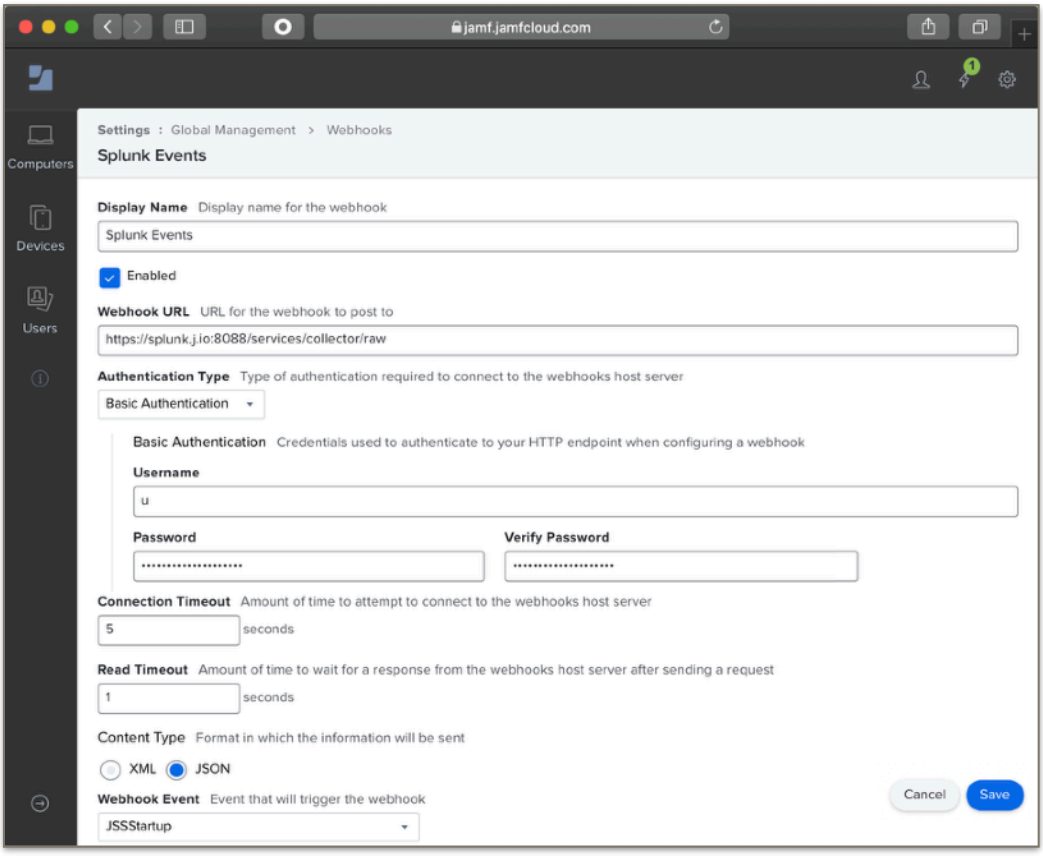
Configuring Jamf Pro to send Webhooks

Webhooks are configured in Jamf Pro's Settings > Global Management screen. Click the "New" button to add one...

The display name can be anything you'd like. The webhook URL is the full HTTPS URL where your Splunk HEC is listening. Authentication type will be "Basic". The user name doesn't matter -- Splunk will completely ignore it. The password is the HEC application token Splunk provided when you created an HEC entry for Jamf Pro.

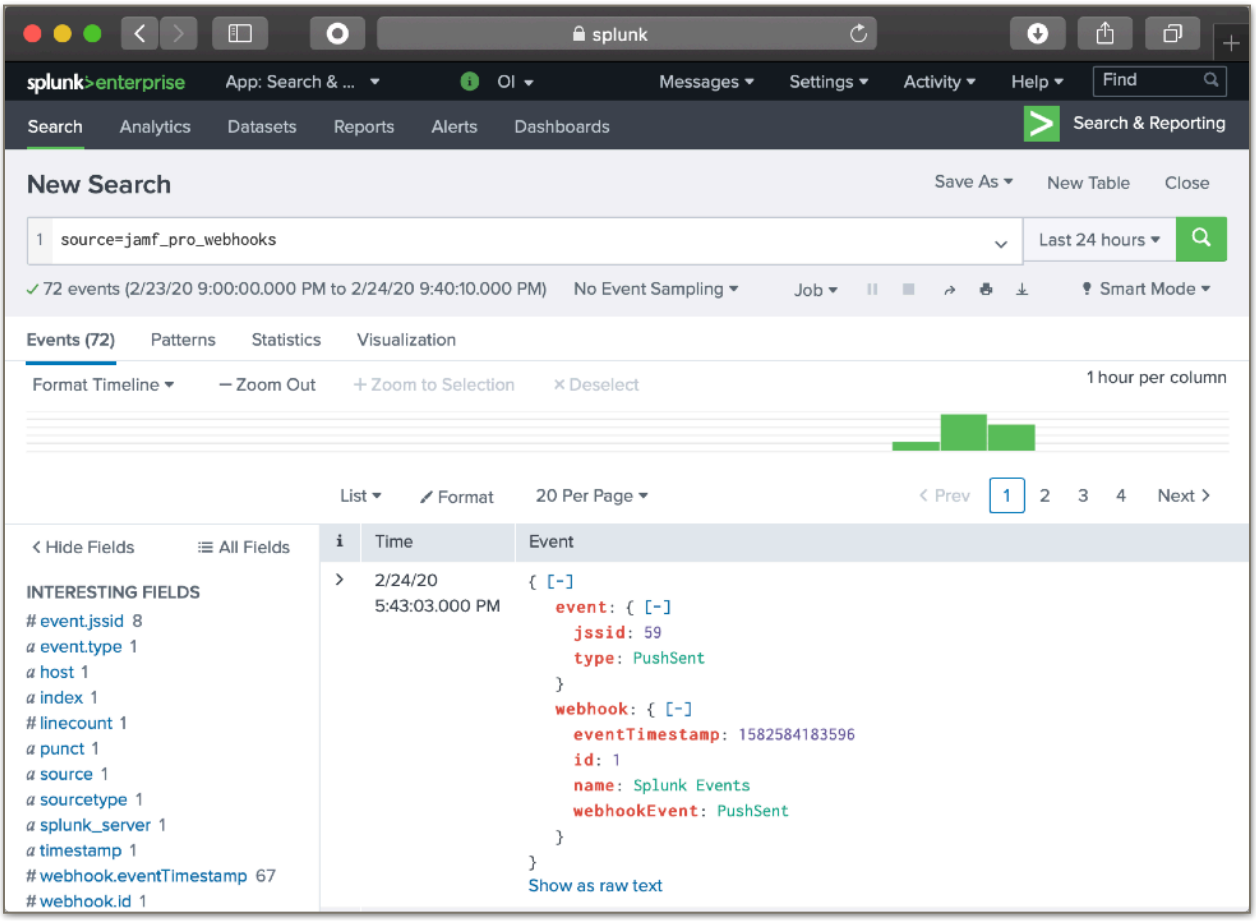
The connection and reply timeouts shouldn't be set too high... some of the webhook types will generate a lot of events and each consumes a thread on the Jamf Pro application server while it waits for Splunk to connect and reply. If you see timeouts in your Jamf Pro logs because your Splunk HEC box is always overloaded with connections, you might want to add a new collector host to your Splunk infrastructure.

Splunk handles both XML and JSON well, but JSON creates less raw data, so it's usually preferred. You can choose any webhook event that you want sent to Splunk.



Click the "Save" button when you've finished entering the information. Once you have the first webhook setup for your Splunk HEC, you can clone it to create additional records for other Jamf Pro events you'd like to send.

Soon, the Jamf Pro server will start sending events to Splunk. You'll know because the record count will start going up in Splunk's source details list on the search screen and you can start querying the data...



Remember that the advantage of webhooks lies in their ability to give us a "live" as-it-happens view of things. So you could ingesting ComputerPolicyFinished or PushSent events to graph throughput in real-time. Or you could use Splunk's "Alerts" features to create notifications of JSSShutdown and JSSStartup events.

Smart group membership change can be a useful tool... you can set up a smart group for a situation that requires a fast response and use the SmartGroupComputerMembershipChange or SmartGroupMobileDeviceMembershipChange events to get that information over to Splunk as soon as it happens.

Appendix 1: Installing Splunk

Overview

> These instructions are for those trying Splunk for the first time. Those using Splunk Cloud or who already have Splunk up and running won't need these instructions.

Splunk is available as a SaaS solution, and also as an "Enterprise" version for self hosting. A free tier is available for evaluation or limited-scale use. Splunk Cloud and Splunk Enterprise are similar in everyday use, but there are some differences in interface and features. Splunk's documentation does a good job of describing product differences where they occur.

This section describes a simple (single server) installation of the self-hosted version of Splunk installed on an Ubuntu host. It may be considered as a lab experiment for those new to the platform. Others may prefer to use the cloud version and skip the installation steps. Splunk is also available for Mac and Windows, so follow the Splunk Installation Guide if you're using one of those options.

Installing Splunk

Sign up for a Splunk account and download the .deb installer package from https://www.splunk.com/en_us/download/splunk-enterprise.html. The terminal session below shows the steps to copy the installer over to the server, install it, and run the initial setup.

```
# scp the file from your computer to the server...
$ scp me@yoursplunkhost.your.org /Users/Me/Downloads/splunk-8.0.1-6db-linux-2.6-amd64.deb

# ssh to splunk server:
$ ssh me@yoursplunkhost.your.org
Welcome to Ubuntu 18.04.3 LTS (GNU/Linux 4.15.0-1058-aws x86_64)
Last login: Wed Jan 29 22:54:04 2020 from 194.223.164.114

$ ls -l
-rw-r--r-- 1 ubuntu ubuntu 354176946 Feb  3 17:08 splunk-8.0.1-6db836e2fb9e-linux-2.6-amd64.deb

$ sudo dpkg -i splunk-8.0.1-6db836e2fb9e-linux-2.6-amd64.deb
Selecting previously unselected package splunk.
(Reading database ... 83930 files and directories currently installed.)
Preparing to unpack splunk-8.0.1-6db836e2fb9e-linux-2.6-amd64.deb ...
Unpacking splunk (8.0.1) ...
Setting up splunk (8.0.1) ...
complete

$ cd /opt/splunk/bin/
$ sudo ./splunk enable boot-start
SPLUNK SOFTWARE LICENSE AGREEMENT
[...]
Do you agree with this license? [y/n]: y

Splunk software must create an administrator account during startup. Otherwise, you cannot log in.
Please enter an administrator username: <secret>
Please enter a new password: <secret>
Please confirm new password:
```

```
Copying '/opt/splunk/etc/openldap/ldap.conf.default' to '/opt/splunk/etc/openldap/
ldap.conf'.
Generating RSA private key, 2048 bit long modulus
.....................+++++
..........+++++
e is 65537 (0x10001)
writing RSA key

Generating RSA private key, 2048 bit long modulus
............+++++
.......................+++++
e is 65537 (0x10001)
writing RSA key

Moving '/opt/splunk/share/splunk/search_mrsparkle/modules.new' to '/opt/splunk/share/
splunk/search_mrsparkle/modules'.
Init script installed at /etc/init.d/splunk.
Init script is configured to run at boot.

$ sudo service splunk start

# Verify running...
$ curl http://localhost:8000
<!doctype html><html><head><meta http-equiv="content-type" content="text/html;
charset=UTF-8"><meta http-equiv="refresh" content="1;url=http://localhost:8000/en-
US/"><title>303 See Other</title></head><body><h1>See Other</h1><p>The resource has
moved temporarily <a href="http://localhost:8000/en-US/">here</a>.</p></body></html>

# Optional: Setup firewall so other hosts in vpc/subnet don't have open access...
$ sudo ufw allow ssh         # Don't skip or you'll be locked out!
$ sudo ufw allow 8000/tcp    # Enable Splunk Web app http...
$ sudo ufw allow 8088/tcp    # Enable Splunk Event Listener http...
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
$ sudo ufw default deny
Default incoming policy changed to 'deny'
```

You can test firewall/proxy/dns setup with `http://<fqdn>:8000` from a browser, but don't submit the admin password you created when installing the software... You're still running on HTTP.

Appendix 2: Configuring Splunk for SSL

Overview

Splunk will create a self-signed SSL certificate when it is first installed. That's good enough to get initial web access if you care to accept an untrusted cert into your computer's keystore, but we'll need to install a new server identity file on the Splunk server if we want to allow other systems to make secure connections.

In this example we're using Let's Encrypt as an identity source, but you may prefer an identity signed by your enterprise CA or by a third party CA if that is your standard practice. The directory where Splunk certificates are typically stored and the configuration file changes needed to tell Splunk to use them are the same in all of these cases.

Install Let's Encrypt to get an SSL Cert

```
# STEP 1: Open port 80... the the Let's Encrypt certbot runs a temp web server to
# publish a challenge password. Let's Encrypt uses it to verify server identity.
$ sudo ufw allow 80/tcp

# STEP 2: Install
# Apt-get method...
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository universe
$ sudo add-apt-repository ppa:certbot/certbot
$ sudo apt-get update
$ sudo apt-get install certbot
$ sudo certbot certonly --standalone -d yoursplunkhost.your.org

# Or you can use the EFF Method...
$ sudo wget https://dl.eff.org/certbot-auto -O /usr/sbin/certbot-auto
$ sudo chmod a+x /usr/sbin/certbot-auto
$ sudo certbot-auto certonly --standalone -d yoursplunkhost.your.org

# The Let's Encrypt installer will ask you to accept the license, provide a
notification email, and if you want to get marketing emails.

# Generated certs are saved in /etc/letsencrypt/live/yoursplunkhost.your.org/.

# Copy the public trust chain and private key to a dir where splunk will read them...
$ sudo mkdir /opt/splunk/etc/mycerts
$ sudo chown -R splunk:splunk /opt/splunk/etc/mycerts
$ sudo cp /etc/letsencrypt/live/yoursplunkhost.your.org/fullchain.pem /opt/splunk/etc/
mycerts
$ sudo cp /etc/letsencrypt/live/yoursplunkhost.your.org/privkey.pem   /opt/splunk/etc/
mycerts
$ sudo chown -R splunk:splunk /opt/splunk/etc/mycerts
```

Auto-renew Let's Encrypt Certs

```
$ sudo touch /etc/letsencrypt/renewal-hooks/deploy/splunk.sh
$ sudo chmod +x /etc/letsencrypt/renewal-hooks/deploy/splunk.sh
$ sudo nano /etc/letsencrypt/renewal-hooks/deploy/splunk.sh
```

Script:

```
#!/bin/bash
# Save in /etc/letsencrypt/renewal-hooks/deploy/splunk.sh
# This will be run automatically by certbot every time it completes a renewal
set -e
for domain in $RENEWED_DOMAINS; do
  case $domain in
  yoursplunkhost.your.org)
    # Make sure the certificate and private key files are
    # never world readable, even just for an instant while
    # we're copying them into daemon_cert_root.
    umask 077
    cp /etc/letsencrypt/live/yoursplunkhost.your.org/fullchain.pem /opt/splunk/etc/
mycerts/fullchain.pem
    cp /etc/letsencrypt/live/yoursplunkhost.your.org/privkey.pem /opt/splunk/etc/
mycerts/privkey.pem
    chown -R splunk:splunk /opt/splunk/etc/mycerts
    sudo /opt/splunk/bin/splunk restart >/dev/null
    ;;
  esac
done
```

Configure Splunk to Use our New Identity

```
# CONFIGURE SPLUNK WEB TO USE THE LET'S ENCRYPT CERT

# Grab a copy of splunk's web config template to use as a starting point...
$ sudo cp /opt/splunk/etc/system/default/web.conf /opt/splunk/etc/system/local/
web.conf
$ sudo chown splunk:splunk /opt/splunk/etc/system/local/web.conf

# Edit the duplicate copy of web.conf we just created.
# Near the top there's a [settings] stanza.
# - Change enableSplunkWebSSL to true.
# - A little further down  two other lines to specify the certs to use.

$ sudo nano /opt/splunk/etc/system/local/web.conf

[settings]

# this determines whether to start SplunkWeb in http or https.
# DEFAULT: enableSplunkWebSSL = true
enableSplunkWebSSL = true

# SSL certificate files.
# DEFAULT: privKeyPath = $SPLUNK_HOME/etc/auth/splunkweb/privkey.pem
# DEFAULT: serverCert = $SPLUNK_HOME/etc/auth/splunkweb/cert.pem

serverCert = /opt/splunk/etc/mycerts/fullchain.pem
privKeyPath = /opt/splunk/etc/mycerts/privkey.pem

# After saving the changes to the .conf file, restart splunkd
$ sudo /opt/splunk/bin/splunk restart

# Test connection to Splunk via a web browswer and check the Cert. If good, you can
now log into the console safely.
```

Appendix 3: Configuring Splunk to Receive Event Data

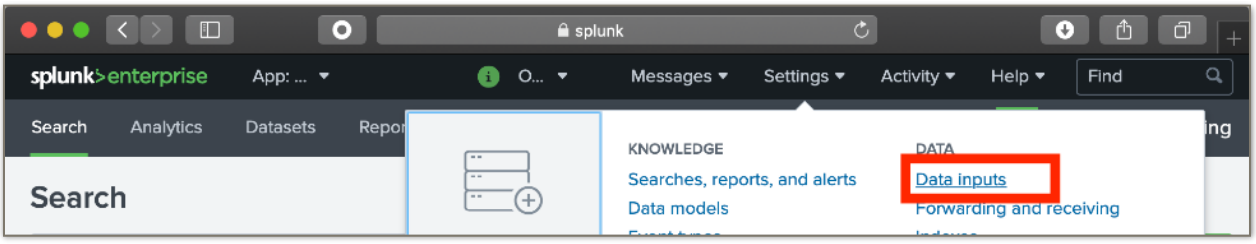
Configure Splunk to listen for events with an HTTP Event Connector (“HEC”)

It’s possible to do all of these things on the command line or in .conf configuration files, but the web GUI is easier for beginners. Choose the method you prefer. References:
https://docs.splunk.com/Documentation/Splunk/8.0.1/Data/UseHECusingconffiles
https://docs.splunk.com/Documentation/Splunk/8.0.1/Data/UsetheHTTPEventCollector

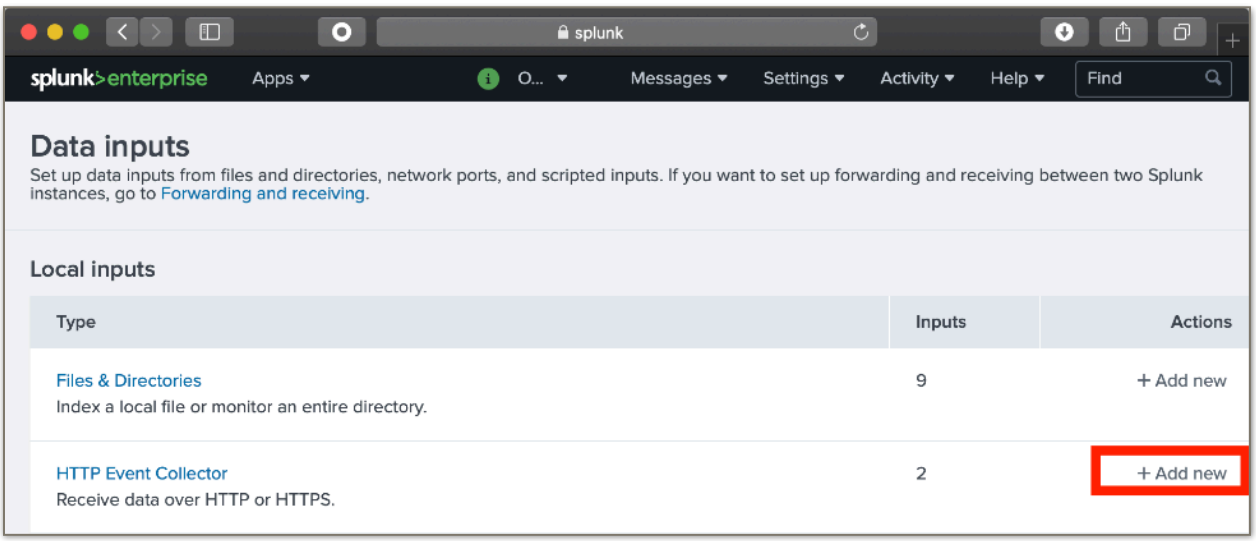
Jamf Protect and Jamf Pro webhooks can both transmit data to Splunk as HTTPS posts. We'll need to configure Splunk to receive these messages to collect them. If you're collecting both Jamf Pro webhooks and Jamf Protect data, you would add two separate HEC data inputs, one for each. That way, each will have its own authentication token and its events will be listed as coming from separate sources in the Splunk database.

As we add each HEC entry, Splunk will provide an authentication token. We'll use this when configuring Jamf applications to send information to Splunk.

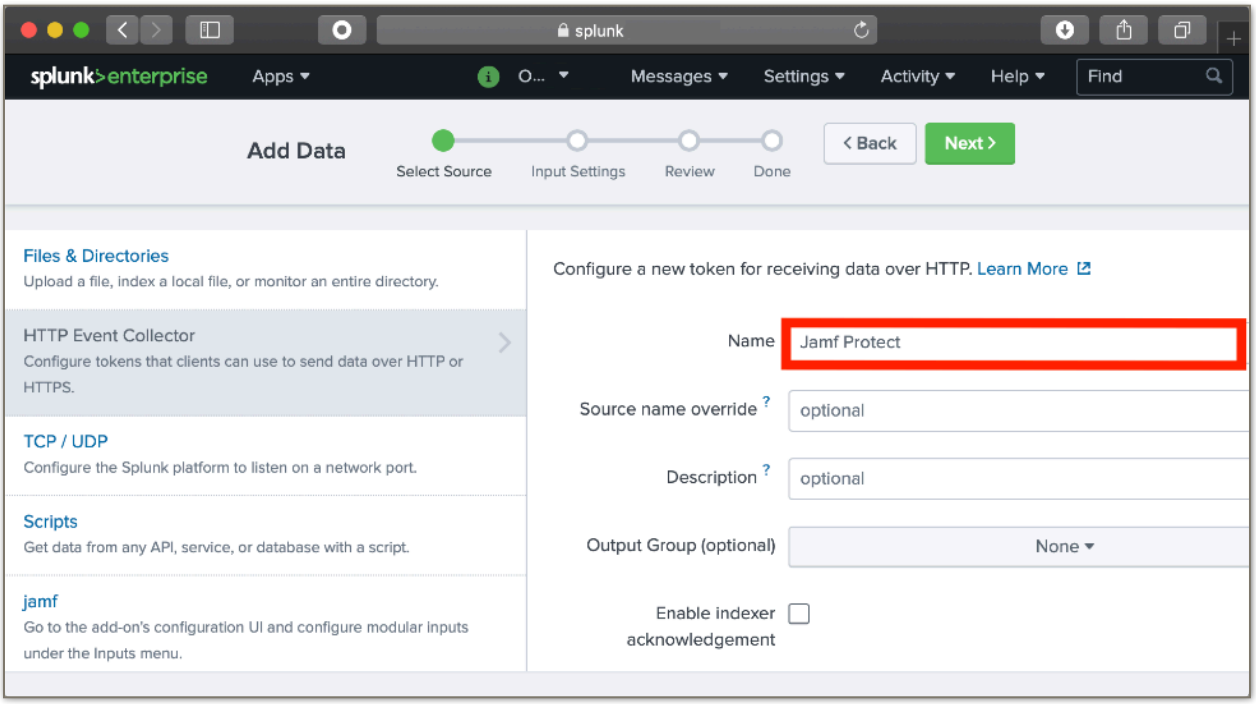
Open SplunkWeb in your browser. E.g. https://yoursplunkhost.your.org:8000. Login using the administrator account and select the “Settings” drop-down menu, then “Data > Data Inputs”...



Click the “HTTP Event Collector > Add new” option...



In the setup wizard, name the collector. In this example, we are setting a collector for Jamf Protect, but if you're adding a collector for Jamf Pro webhooks, you'd name it something different. Otherwise the process for setting up the two collectors is the same. If you don't provide a "Source name override", Splunk will use the collector's name. If you want something else, enter the override.

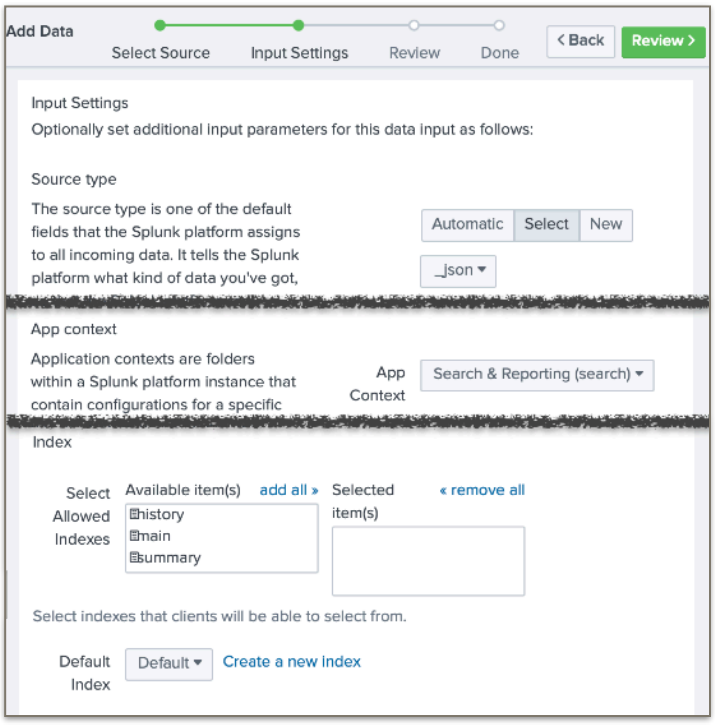


The default values on the input settings page will work fine so it's not critical you change anything here.

The Automatic source type works because both Protect and webhooks use JSON which is easily detected by Splunk. But we can save Splunk some work by explicitly setting "Source type" to **_json**.

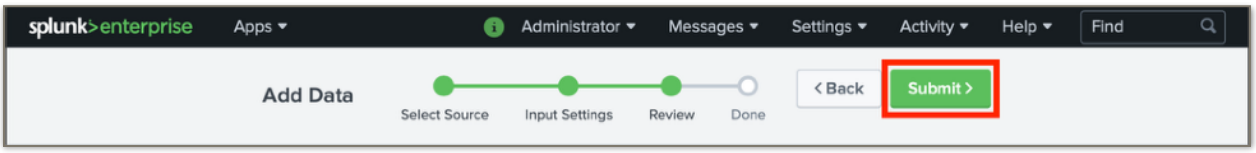
The App Context should be set to Search & Reporting.

The default index setting is used unless you have a more advanced Splunk setup with distributed indexers. When a default Splunk setup is being used, the events will be indexed into "main" but you don't have to select it as an allowed index when using the defaults.

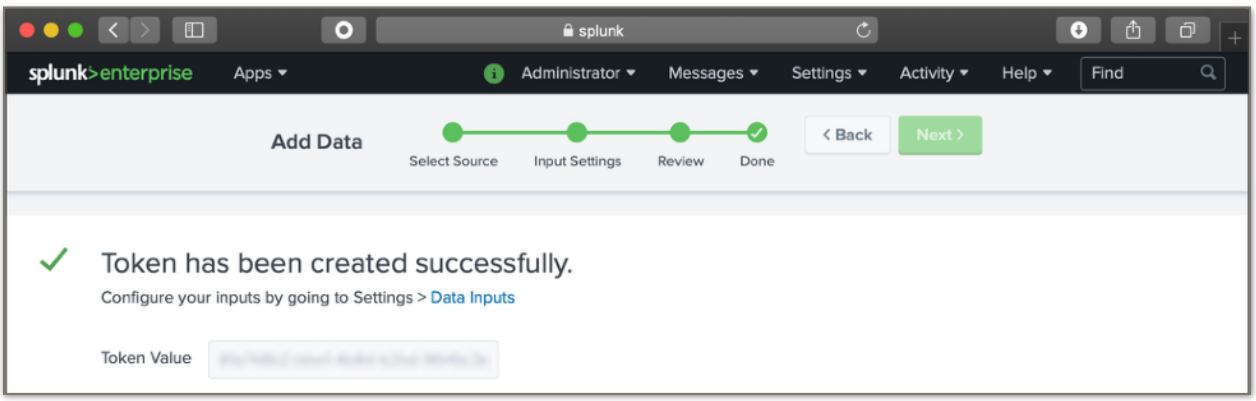


Click the “Review“ button at the top to continue.

Review your settings and click the “Submit“ button.

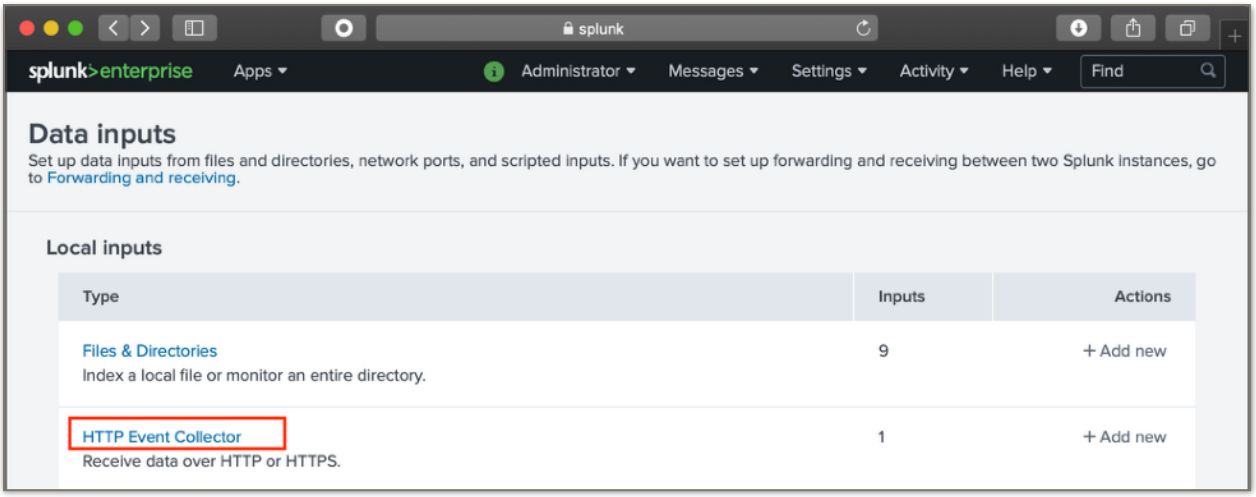


In the final setup page, note the Token Value provided by Splunk. We’ll need this when configuring Jamf Protect or Jamf Pro webhooks to send data to Splunk.

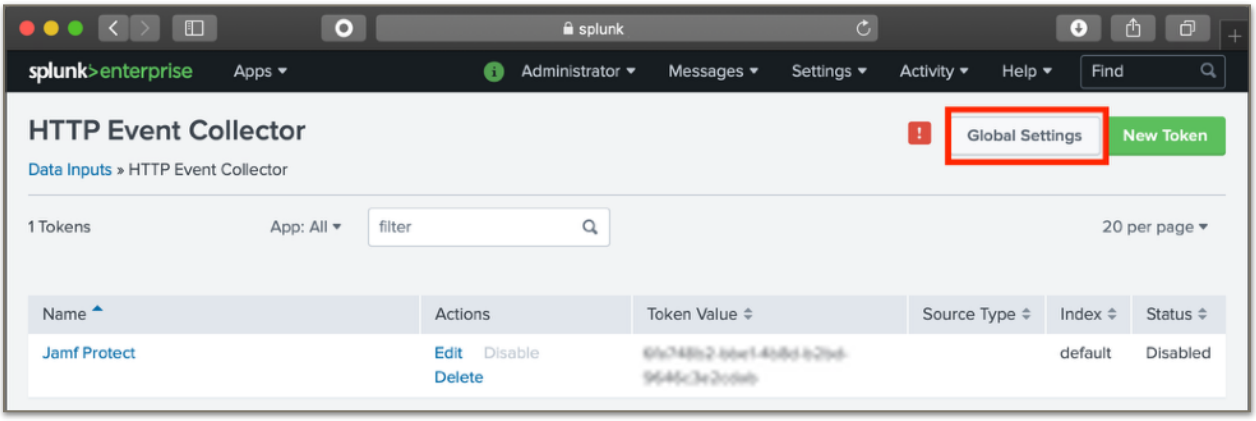


Set the HTTP Event Listener "Default Settings"

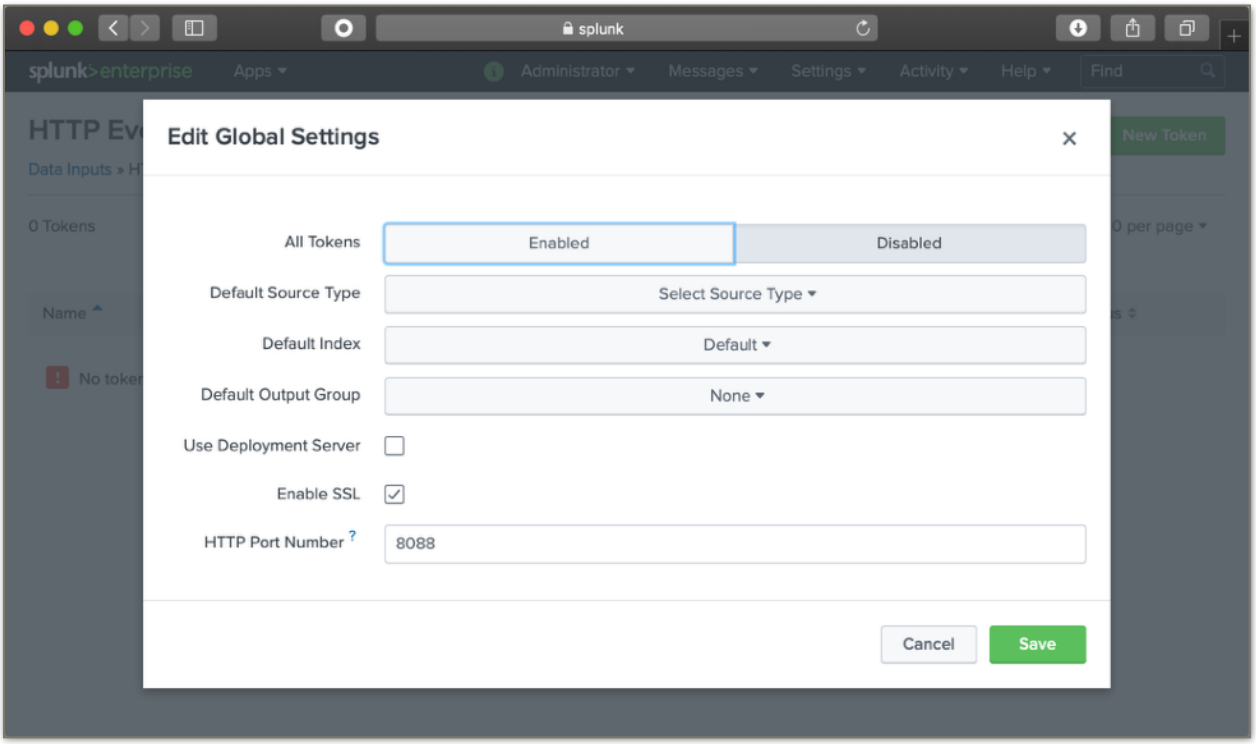
Return to Settings > Data > Data Inputs > “HTTP Event Collector”.



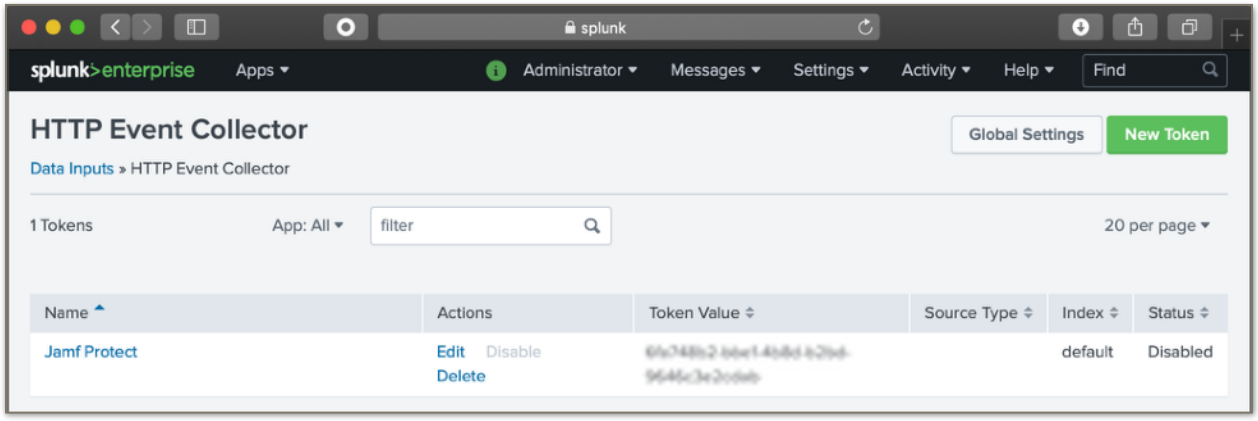
If you're working on a new Splunk install, the Collector we just added is not yet enabled. You'll see a red "!" at the top of the collector list screen. If you click it, you'll be warned that no collector is currently enabled. To fix that, click the "Global Settings" button…



Splunk's default settings are fine as they are but if you want, you can set the listing port you prefer as long as it's not already in use by some other service and it's going to get through any firewalls and proxies in your environment. The default HTTP Port Number is 8088 for self-hosting and 443 for Splunk Cloud. Note the HTTP Port number -- we'll need it when we configure Jamf Protect Actions and Jamf Pro Webhooks. Click the "Save" button when you're finished.



The Collectors list now shows the Jamf Connect Connector as enabled.



Open communications to the HTTP Event Connector

We'll be configuring the Jamf Protect agent running on computers and the Jamf Pro server's webhooks to send event their data to the Splunk server on the port you specified in Splunk Global Settings. Configure any firewalls and proxies in your network to allow the traffic. For example, to allow the connection through an Ubuntu firewall…

```
$ sudo ufw allow 8088/tcp
Rule added
Rule added (v6)
```

Set the TLS certificate for the HTTP Event Connector

We'll want to send Jamf Protect events and Jamf Pro webhooks over an authenticated connection. They may or may not contain sensitive information but they will contain an auth header and we wouldn't want that traveling unencrypted. You'll probably want to use a third-party/Public CA certificate so it's pre-trusted by sending hosts. If you were to use a self-signed certificate, you'd have to install the trust chain on every host that needs to connect to your Splunk HEC.

If you are using Splunk Cloud, these certificates and settings are already configured by Splunk. If you host your own Splunk, we'll need to tell it to use something other than it's own self-signed cert.

In this example, we're going to run the HEC on the same server as the SplunkWeb app. We've already set up Let's Encrypt to create the certs for SplunkWeb so we'll just tell the HEC it can use that cert too.

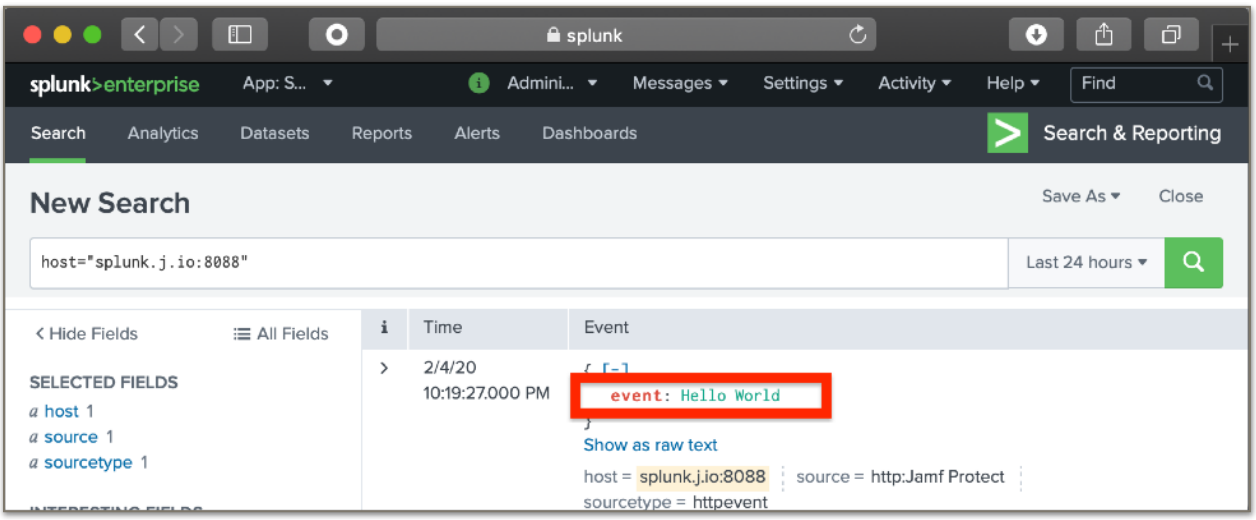
```
# CONFIGURE SPLUNK HTTPS EVENT COLLECTOR TO USE THE LET'S ENCRYPT CERT
# Make a backup copy of the existing listener config file

$ sudo cp /opt/splunk/etc/apps/splunk_httpinput/local/inputs.conf \
/opt/splunk/etc/apps/splunk_httpinput/local/inputs.conf.bak
# Edit the duplicate copy of inputs.conf we just created.
$ sudo nano /opt/splunk/etc/system/local/inputs.conf

# There are lots of things that can be configured here. See the Splunk docs.
# The required items and sole contents of my version of this file are as follows:
[http]
disabled = 0
serverCert = /opt/splunk/etc/mycerts/fullchain.pem
privKeyPath = /opt/splunk/etc/mycerts/privkey.pem
```

```
# You can also bundle your private key and public cert chain in one .pem file, or with
the public cert chain in a .pem and the private key in a .key file if your CA provided
it in that way, or if you used openssl to convert it to that format. This is the
config if using separate public .pem and private .key files where the .key file has a
passphrase...
[http]
disabled = 0
serverCert = /opt/splunk/etc/mycerts/jamfseio.pem
privKeyPath = /opt/splunk/etc/mycerts/jamfseio.key
sslPassword = passphrasetokeyfilecMNjCwk2lUqHPKgrb4Oq0Kfxc=

# After saving the changes to the .conf file, restart splunkd. On Ubuntu...
$ sudo /opt/splunk/bin/splunk restart

Test the HTTP Event Collector (“HEC”)
# HEC TESTS

# CHECK FOR LISTENING PORTS

$ sudo lsof -i -P -n | grep LISTEN | grep splunkd
splunkd   22244            root    4u  IPv4 296287      0t0  TCP *:8089 (LISTEN)
splunkd   22244            root  104u  IPv4 307770      0t0  TCP *:8088 (LISTEN)
splunkd   22244            root  106u  IPv4 296375      0t0  TCP *:8000 (LISTEN)

# CHECK THE CERTIFICATES IN USE FOR TLS

# On the splunk server host, check localhost...
echo | openssl s_client -showcerts -connect localhost:8088 | grep -E '(subject|
issuer)'
# On a remote machine...
echo | openssl s_client -showcerts -connect yoursplunkhost.your.org:8088 | grep -E
'(subject|issuer)'

# This is what you'll see if Splunk is using it's own self-signed certs (wrong)...
depth=1 C = US, ST = CA, L = San Francisco, O = Splunk, CN = SplunkCommonCA,
emailAddress = support@splunk.com
verify error:num=19:self signed certificate in certificate chain
subject=CN = SplunkServerDefaultCert, O = SplunkUser
issuer=C = US, ST = CA, L = San Francisco, O = Splunk, CN = SplunkCommonCA,
emailAddress = support@splunk.com

# If everything's right, you'll see your own cert (we used let's encrypt...)
depth=2 O = Digital Signature Trust Co., CN = DST Root CA X3 / verify return:1
depth=1 C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3 / verify return:1
depth=0 CN = yoursplunkhost.your.org / verify return:1
subject=CN = yoursplunkhost.your.org
issuer=C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
```

Once the certificate is configured and you've set up your HTTP Event Connector(s), you can use collector auth token provided by Splunk to send a test event using an https client like curl, PowerShell, or Postman. This is an example of using curl...

```
# USE CURL TO CHECK NETWORK, AUTH, AND COLLECTOR CONFIGURATION
$ curl https://yoursplunkhost.your.org:8088/services/collector/raw --header
"Authorization: Splunk ZZZZ48b2-Zbe1-Zb8d-Z2bd-9646cZZZZZZZ" --data '{"event": "Hello
World"}'

# This is what you'll see if you use the wrong auth token...
<?xml version="1.0" encoding="UTF-8"?>
<response><messages><msg type="WARN">call not properly authenticated</msg></messages>
</response>
# This is the response if you've got everything configured properly...
{"text":"Success","code":0}
```

You can also do a search on the collector host and see the record as it appears in Splunk.



APPENDIX 4: Configuring SSO for Splunk

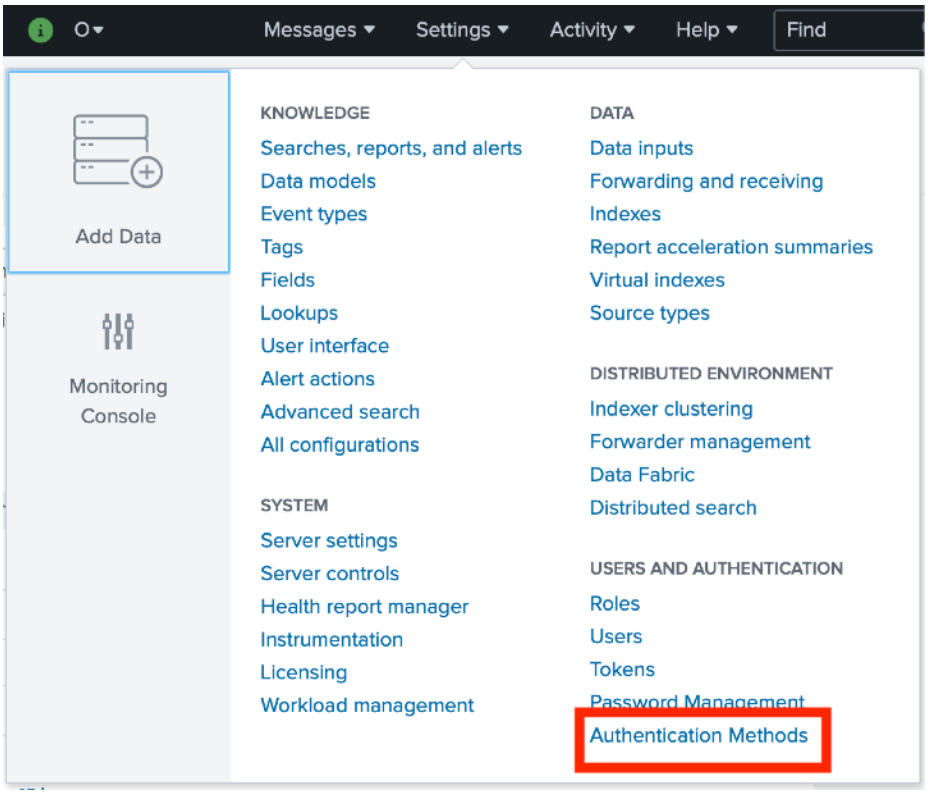
If you want others to view your work, you'll need to configure users. Like Jamf Pro, Splunk allows you to create local user accounts, but if you have an enterprise directory or identity provider, you can take advantage of the users, passwords, and groups already available there. This is an example of setting up SAML SSO using Azure AD, but a similar process is used with any IdP.

Note: It's always a good idea to make a backup before doing any major changes, but if you manage to get yourself locked out of Splunk, the SSO bypass URL is : https://yoursplunkhost.your.org:8000/en-US/account/login?loginType=splunk. Note, however, this page will be disabled as soon as you turn SAML on in Splunk. To turn it back on, you'll need to ssh into the server to edit the server.conf file, setting "enable_insecure_login" to true. Then you can get use the bypass URL above to go back in with a local Splunk administrator account.

Ref: https://docs.splunk.com/Documentation/Splunk/latest/Admin/Webconf

Step 1 - Figure out your access groups.

Create an AD/AzureAD group to identify Splunk admins and one to identify general (non-privileged) users. The names are arbitrary so follow your own group-naming standard. Assign users to these groups as appropriate. We'll need these groups when mapping Azure AD group memberships to Splunk roles.



If you created the groups in AD, you may have to wait for them to sync to Azure AD. Once the new groups sync to Azure AD, you can click on them in Azure AD and get their **Object IDs**. We use the OIDs to identify SAML groups in Splunk... you can't use the group name.

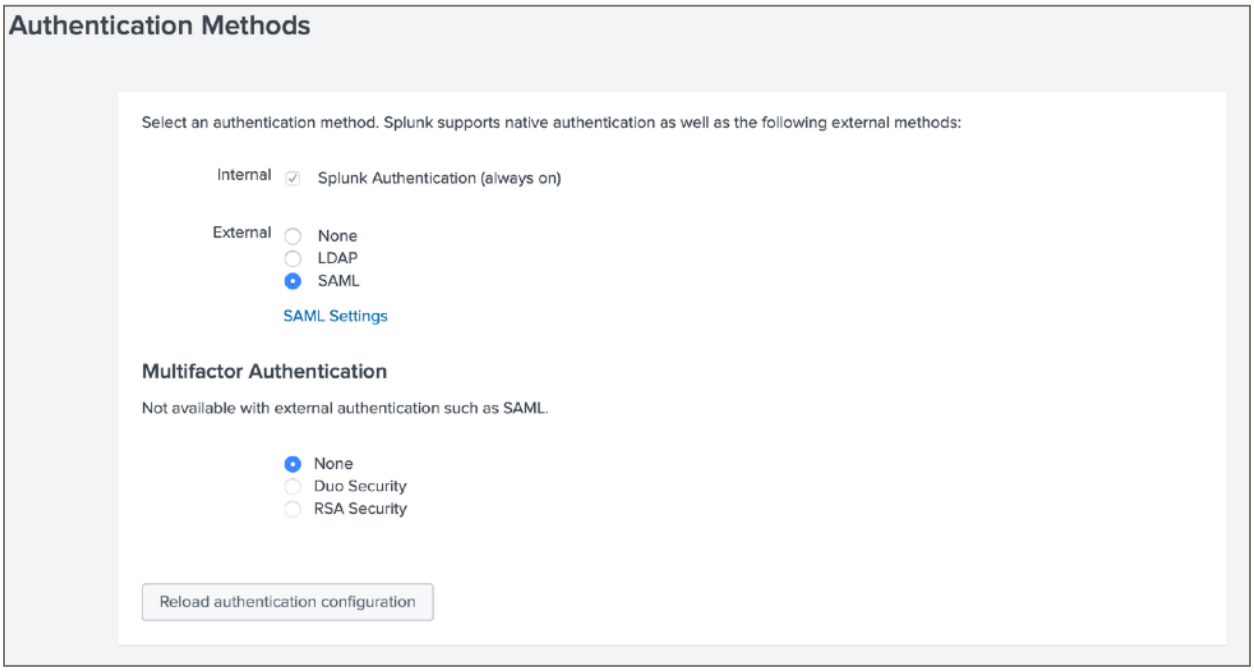
Step 2 - Turn on SAML in Splunk and download the IdP Metadata File

References:

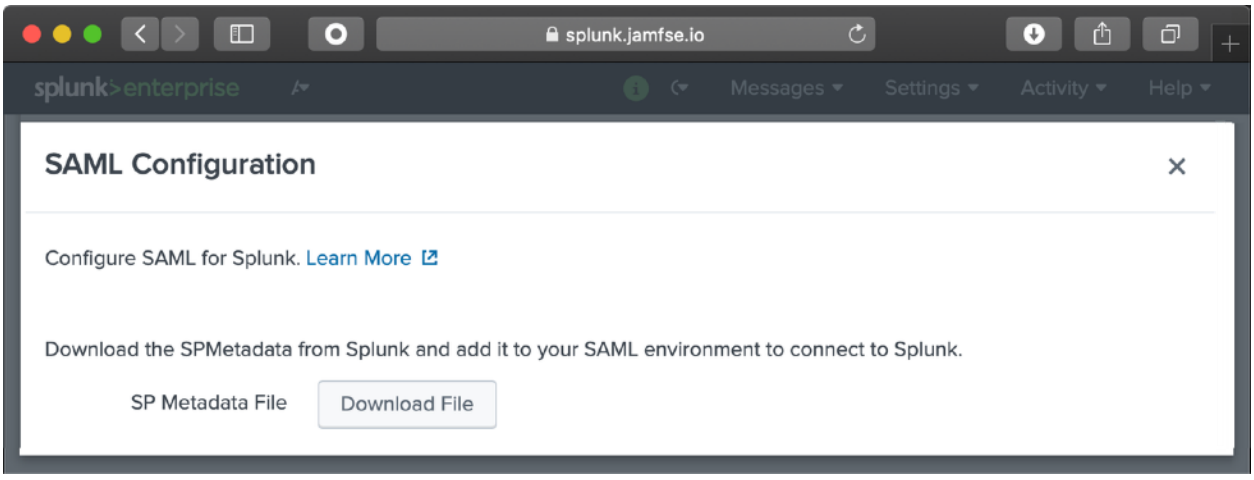
https://docs.splunk.com/Documentation/Splunk/8.0.1/Security/ConfigureSSOAzureADandADFS

https://www.splunk.com/en_us/blog/cloud/configuring-microsoft-s-azure-security-assertion-markup-language-saml-single-sign-on-sso-with-splunk-cloud-azure-portal.html

Login to SplunkWeb. Go to > Settings > Users and Authentication > Authentication Methods and change Authentication Method > External from “None” to “SAML”.



Then click the SAML Settings link. At the top, there's an option to download an SP MetaData File. Download it to your computer. It contains information about our Splunk setup that we’ll need to feed to Azure AD.

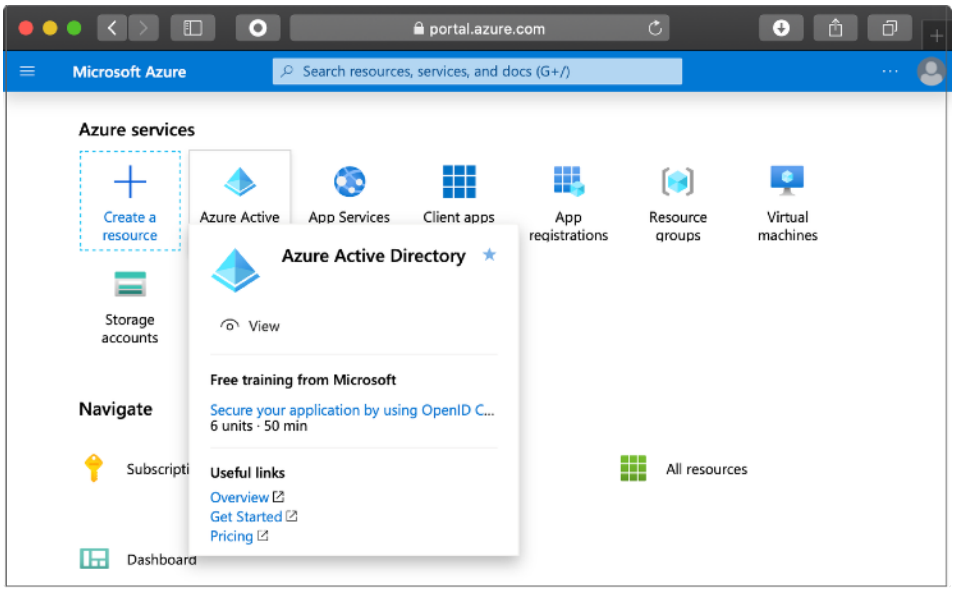


Step 3 - Setup an app in Azure

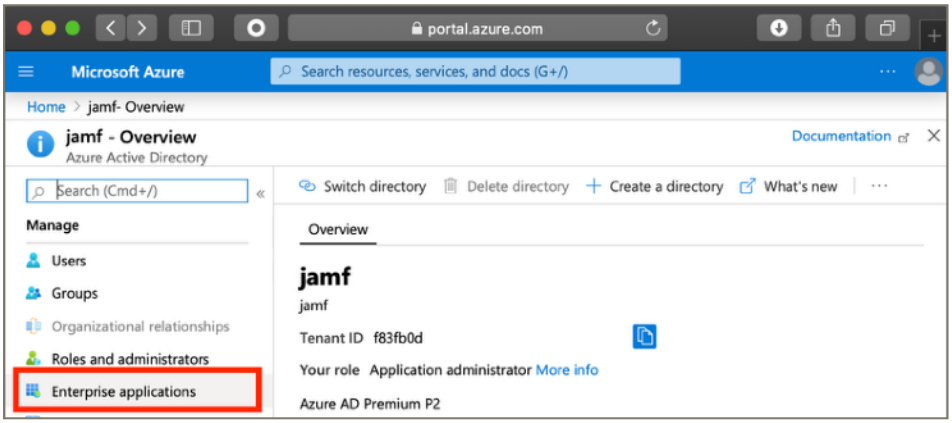
Note: Splunk Enterprise and Splunk Cloud support SP-initiated SSO.

Reference: https://docs.microsoft.com/en-us/azure/active-directory/saas-apps/splunkenterpriseandsplunkcloud-tutorial

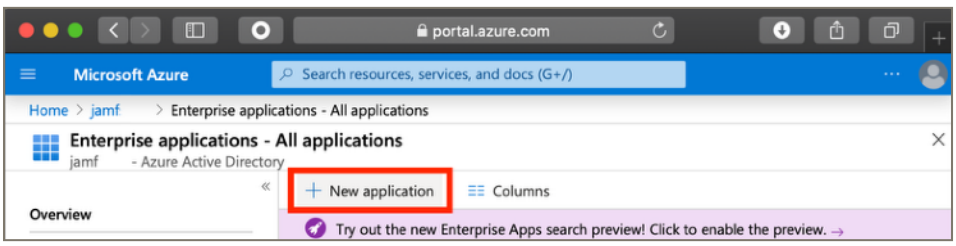
In the Azure portal, go to Azure Active Directory



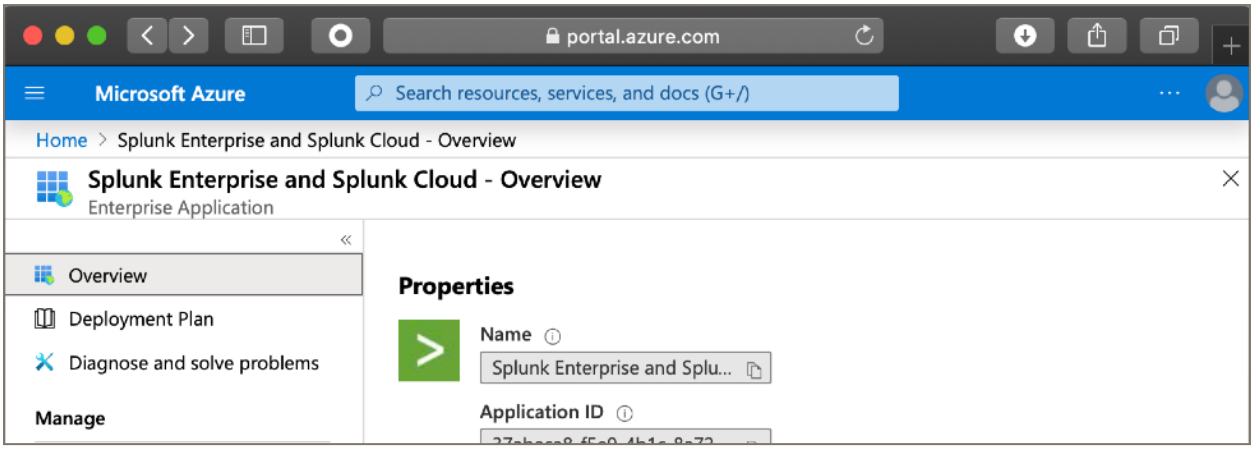
Then to Enterprise Applications



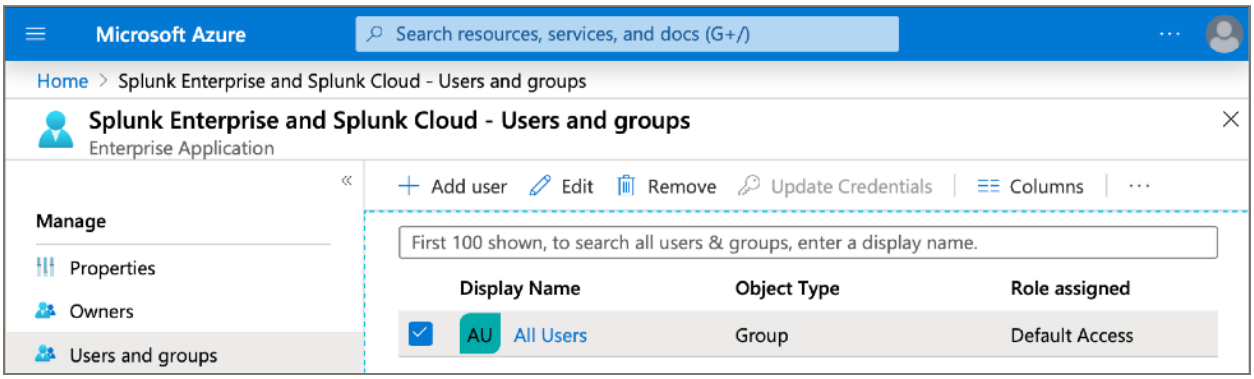
Click the "New Application" button



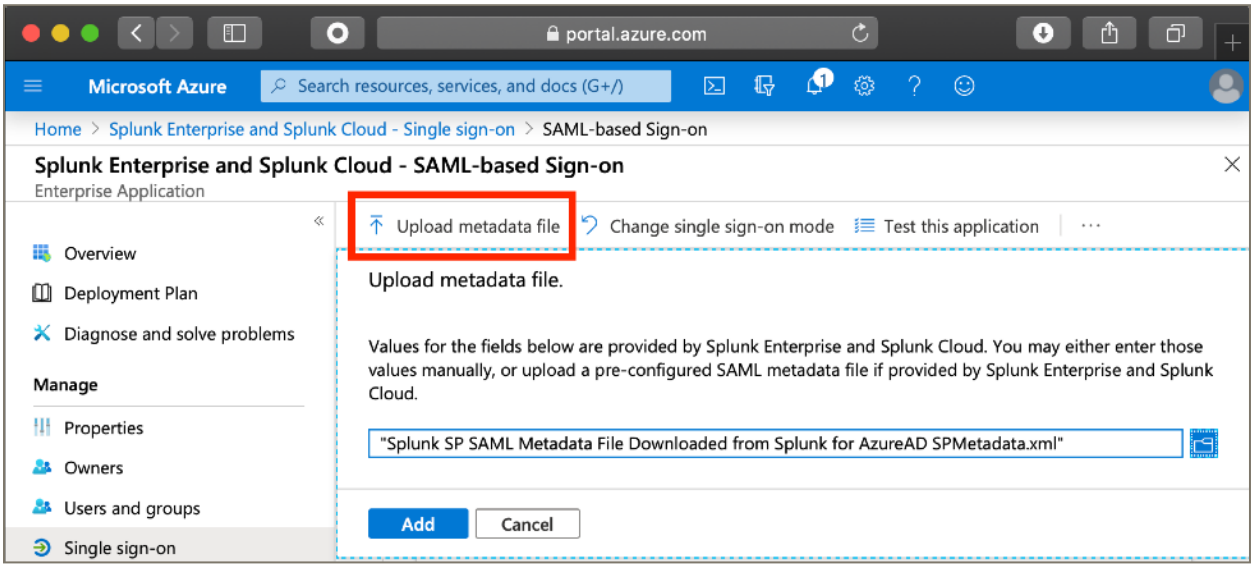
Do a search for “Splunk” and add the **Splunk Enterprise and Splunk Cloud** app.



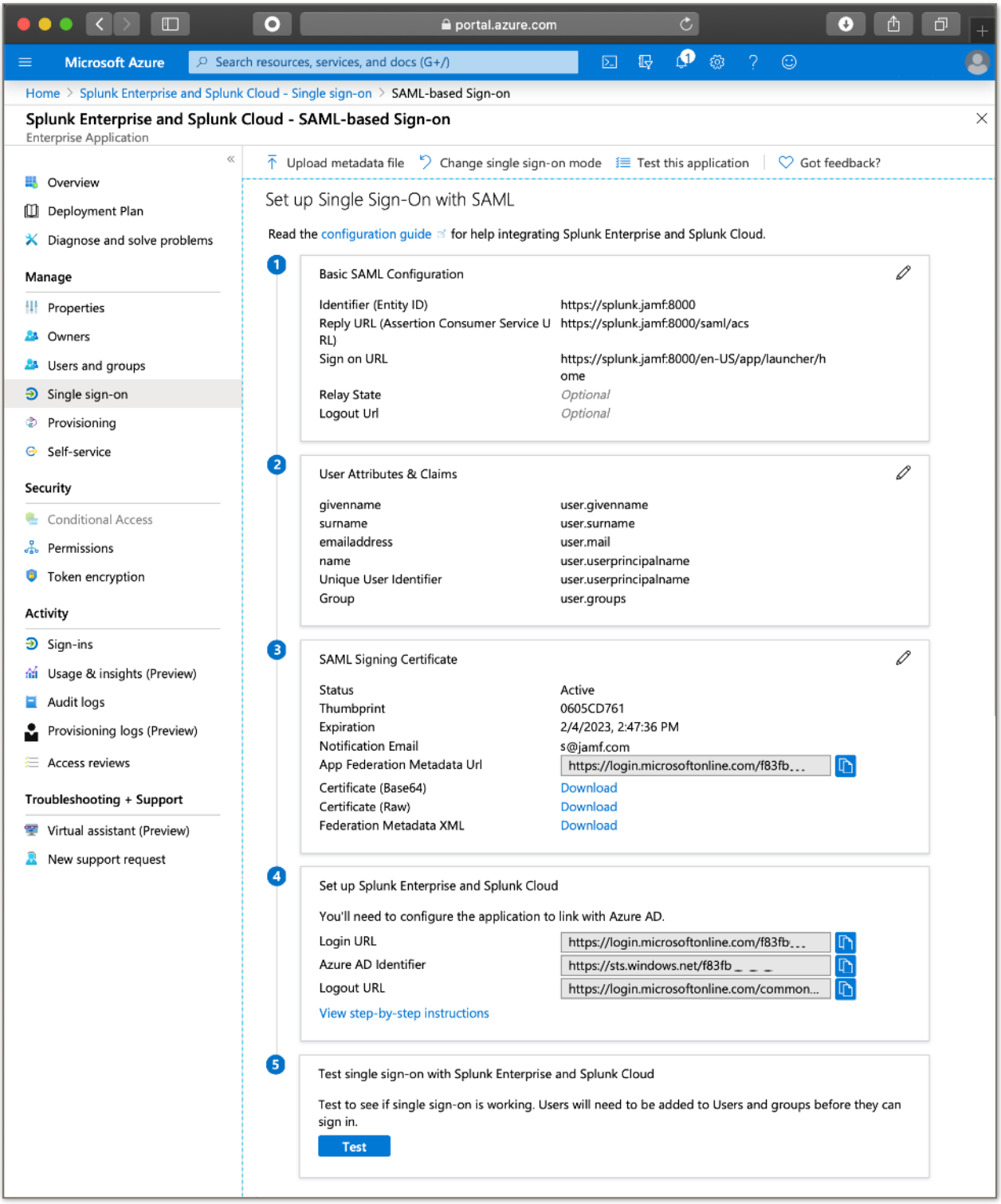
You have to assign the app to users in order for them to access it. In the organization shown below, all users can use Splunk, but you would more typically assign the app to your Splunk Admins and Splunk Users groups.



Go to the Single single-on blade and click "Upload metadata file"...



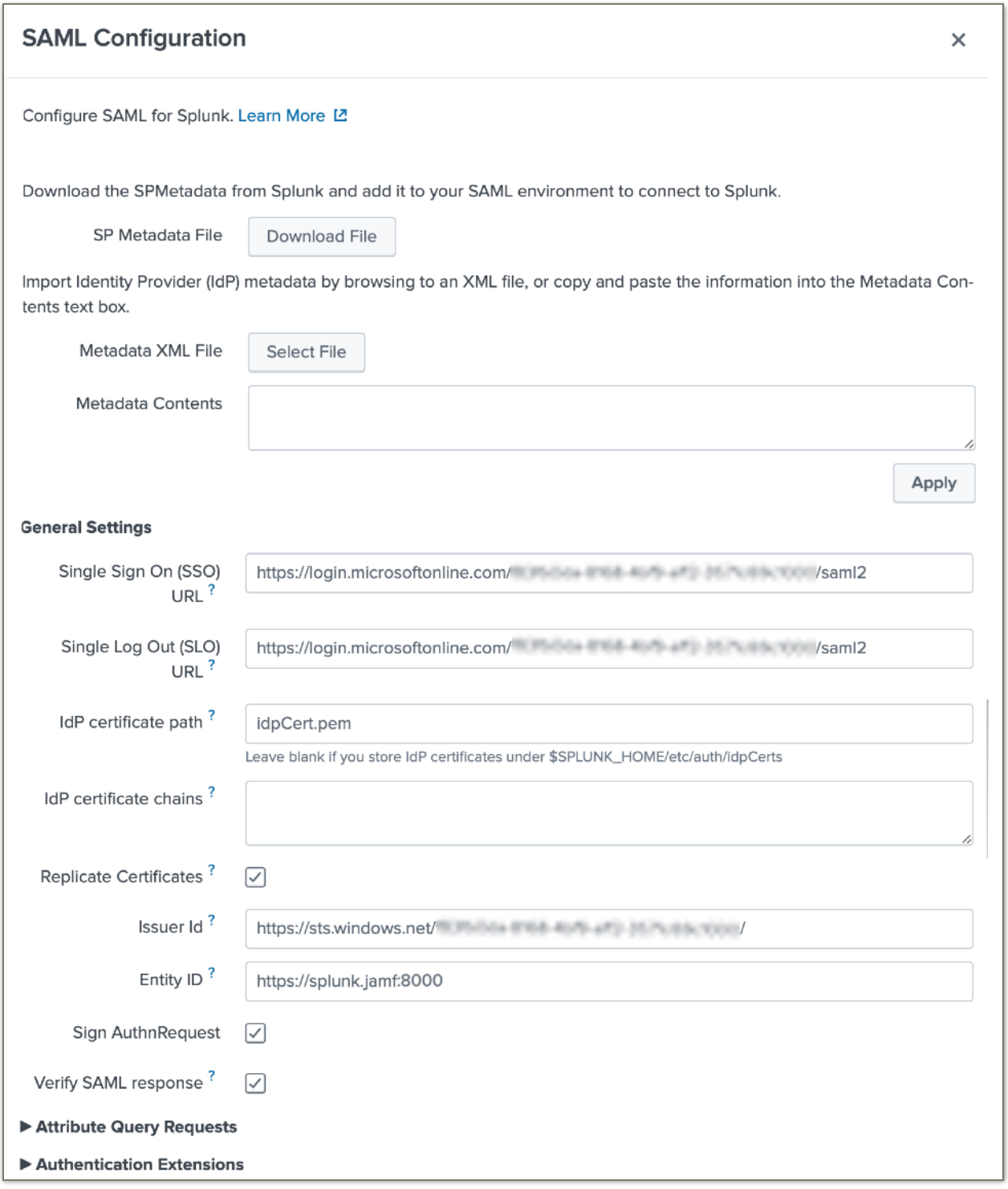
The information from the Splunk-provided metadata file will be filled into the Azure app's SAML settings. Double-check the host name -- Splunk may provide the internal host name rather than the external one.

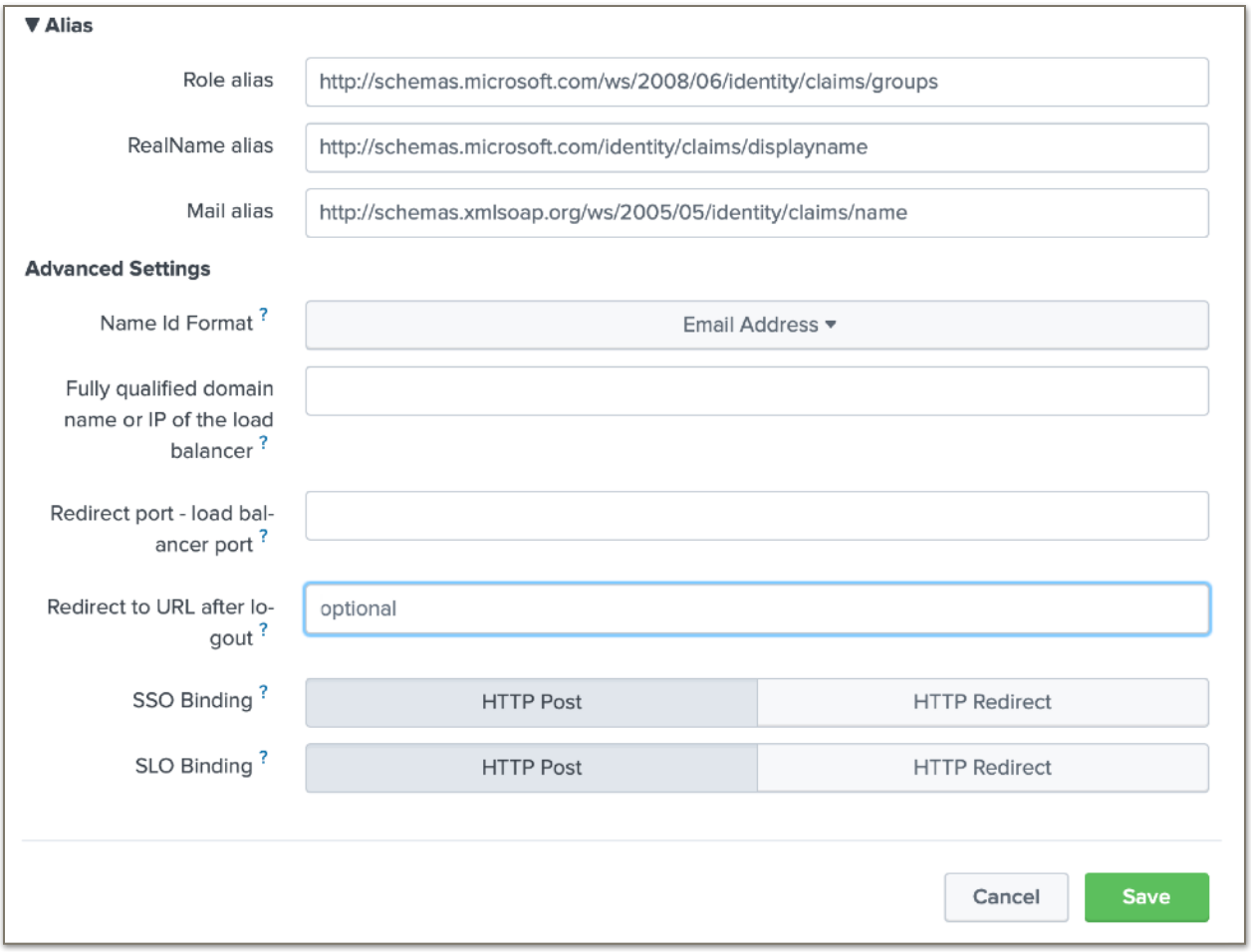


In section 3, there's a Download link next to “Federation Metadata XML“. Download that file. It will help finish some remaining configurations we'll need to make back to Splunk.

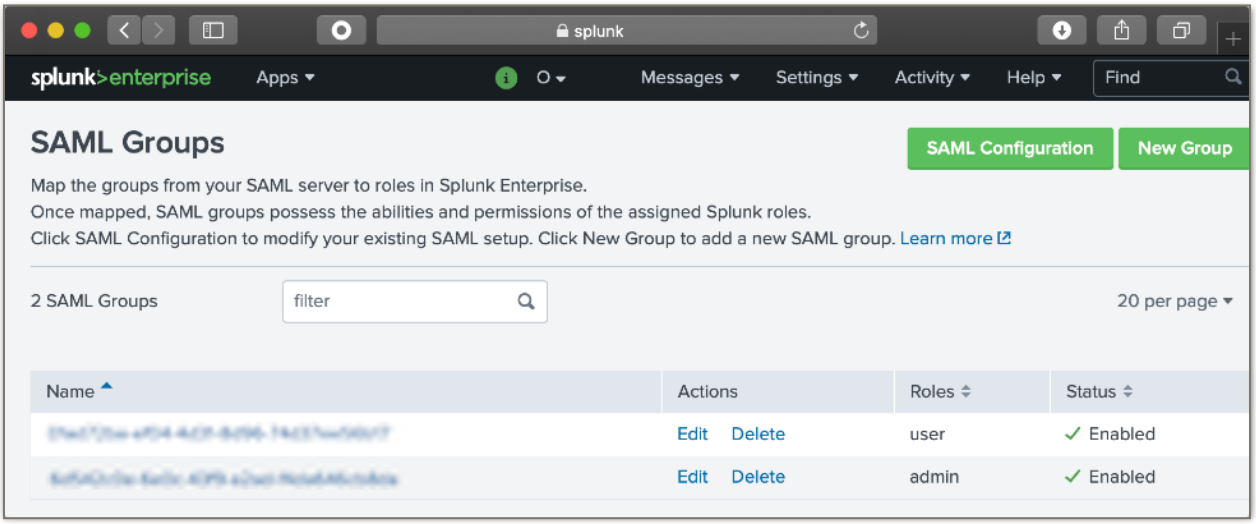
Step 4 - Complete the SAML configuration in Splunk

Back in our SAML config screen in Splunk, the next step is to upload the Metadata XML file we just downloaded from our Azure AD Enterprise App single sign-on configuration screen. That will fill in most of the General Settings section and upload the certificate. You'll need to fill in the Alias information, but in this simplified configuration, the other sections shouldn't require any changes.





Once SAML Configuration is complete, go into the SAML Groups and add the groups you want to use for Admins and Users. Remember that we specify the Azure AD Groups' Object IDs and assign them to their appropriate Splunk roles.



Step 5 - Ready to Test

The Azure AD Enterprise App Single sign-on blade has a test button that simulates what happens if a user selects Splunk from their Azure's Apps dashboard. If you log out of your Azure AD session and go to your Splunk URL, you will see that Splunk's login page is gone and Azure AD's login will appear. Try logging in with a non-admin and an admin Azure AD account. Verify that single-logoff works as well.

Appendix 5: Jamf Pro Webhooks and Data Structures

Note: Please see Jamf's Developer's Portal (https://developer.jamf.com/webhooks) for additional up-to-date information about webhooks.

Overview

The Jamf Pro web app can be configured to emit a webhook on a variety of events. The webhook body will contain two objects in its JSON data: `event` and `webhook`. The basic structure of the webhook object is the same for all webhook event types and includes basic information like when the webhook was sent, the name of the webhook in Jamf Pro, and the event that triggered the webhook.

The following section lists the supported webhook events, how the events are triggered, and samples of the JSON data structures. Some callouts of data that might be helpful when creating reports and alerts is highlighted in yellow.

Webhook Event Details

Computer Events Overview

The ComputerAdded, ComputerCheckin, ComputerInventoryCompleted, ComputerPolicyFinished and ComputerPushCapabilityChanged events return data in the format depicted below, with their respective values for the `webhookEvent` attribute. Each includes details about the Mac in it's event object.


```
{
    "event": {
        "alternateMacAddress": "72:00:01:DD:A0:B9",
        "building": "Block D",
        "department": "Information Technology",
        "deviceName": "John's MacBook Pro",
        "emailAddress": "john.smith@company.com",
        "jssID": 13,
        "macAddress": "60:03:08:A3:64:9D",
        "model": "13-inch Retina MacBook Pro (Late 2013)",
        "osBuild": "16G29",
        "osVersion": "10.12.6",
        "phone": "555-472-9829",
        "position": "Desktop Services Specialist",
        "realName": "John Smith",
        "room": "487",
        "serialNumber": "C02M23PJFH50",
        "udid": "EBBFF74D-C6B7-5599-93A9-19E8BDDEFE32",
        "userDirectoryID": "-1",
        "username": "john.smith"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "name": "Webhook Documentation",
        "webhookEvent": "ComputerPushCapabilityChanged"
    }
}
```

ComputerAdded

This event is triggered when a new computer is enrolled into Jamf Pro. This event will not trigger if a computer is re-enrolled and a record with a matching Hardware UUID already exists within Jamf Pro's inventory.

ComputerCheckIn

This event is triggered when a managed computer reaches out to the Jamf Pro server, to check for tasks. The following device check-in types will trigger the webhook event:

- Startup
- Login
- Logout
- Network State Change
- Enrollment Complete
- Recurring Check-in

The body for this event includes information about the check-in event that triggered the webhook and information about the user that was logged into the device when the check-in occurred is included as part of the `event` object via the `username` key.


```
{
    "event": {
        "computer": {
            "alternateMacAddress": "72:00:01:DD:A0:B9",
            "building": "Block D",
            "department": "Information Technology",
            "deviceName": "John's MacBook Pro",
            "emailAddress": "john.smith@company.com",
            "ipAddress": "10.15.26.202",
            "jssID": 13,
            "macAddress": "60:03:08:A3:64:9D",
            "model": "13-inch Retina MacBook Pro (Late 2013)",
            "osBuild": "16G29",
            "osVersion": "10.12.6",
            "phone": "555-472-9829",
            "position": "Desktop Services Specialist",
            "realName": "John Smith",
            "reportedIpAddress": "10.15.26.202",
            "room": "487",
            "serialNumber": "C02M23PJFH50",
            "udid": "EBBFF74D-C6B7-5599-93A9-19E8BDDEFE32",
            "userDirectoryID": "-1",
            "username": "john.smith"
        },
        "trigger": "CLIENT_CHECKIN",
        "username": "John Smith"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "name": "Webhook Documentation",
        "webhookEvent": "ComputerCheckIn"
    }
}
```

ComputerInventoryCompleted

This event is triggered when a managed computer submits inventory.

ComputerPolicyFinished

This event is triggered when a managed computer executes a policy. Both failed and successfully executed policies will trigger this event. In a future version of Jamf Pro, the response body will be modified to the following, which includes information about the policy that completed:


```
{
    "event": {
        "computer": {
            "alternateMacAddress": "72:00:01:DD:A0:B9",
            "building": "Block D",
            "department": "Information Technology",
            "deviceName": "John's MacBook Pro",
            "emailAddress": "john.smith@company.com",
            "jssID": 6486,
            "macAddress": "60:03:08:A3:64:9D",
            "model": "MacBook Pro (13-inch, 2018)",
            "osBuild": "18D109",
            "osVersion": "10.14.3",
            "phone": "555-472-9829",
            "position": "Desktop Services Specialist",
            "realName": "John Smith",
            "room": "487",
            "serialNumber": "C02M23PJFH50",
            "udid": "EBBFF74D-C6B7-5599-93A9-19E8BDDEFE32",
            "userDirectoryID": "-1",
            "username": "john.smith"
        },
        "policyId": 8,
        "successful": true
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 1,
        "name": "Webhook Documentation",
        "webhookEvent": "ComputerPolicyFinished"
    }
}
```


ComputerPushCapabilityChanged

This webhook event is triggered when there is a change in a device's ability to receive push notifications.

ComputerPatchPolicyCompleted

This event is triggered when a Computer Patch Policy is completed. It contains detailed information about the computer that is running patch and the status of the update attempt on the Mac.


```
{
    "event": {
        "computer": {
            "alternateMacAddress": "72:00:01:DD:A0:B9",
            "building": "Block D",
            "department": "Information Technology",
            "deviceName": "John's MacBook Pro",
```

```
            "emailAddress": "john.smith@company.com",
            "jssID": 13,
            "macAddress": "60:03:08:A3:64:9D",
            "model": "13-inch Retina MacBook Pro (Late 2013)",
            "osBuild": "16G29",
            "osVersion": "10.13.4",
            "phone": "555-472-9829",
            "position": "Desktop Services Specialist",
            "realName": "John Smith",
            "room": "487",
            "serialNumber": "C02M23PJFH50",
            "udid": "EBBFF74D-C6B7-5599-93A9-19E8BDDEFE32",
            "userDirectoryID": "-1",
            "username": "john.smith"
        },
        "deployedVersion": "66.0.3359.117",
        "eventActions": {
            "action": [
                "Executing Patch Policy Google Chrome",
                "Mounting MasterDP",
                "Copying GoogleChrome.dmg...",
                "Installing GoogleChrome.dmg...",
                "Mounting GoogleChrome.dmg...",
                "Mount successful",
                "Installing package contents...",
                "Successfully installed package contents",
                "Unmounting GoogleChrome.dmg...",
                "Unmount successful"
            ]
        },
        "patchPolicyId": 4,
        "patchPolicyName": "Chrome Test",
        "softwareTitleId": 2,
        "successful": true
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 5,
        "name": "Webhook Documentation",
        "webhookEvent": "ComputerPatchPolicyCompleted"
    }
}
```


DeviceAddedToDEP

This event is triggered when Jamf Pro receives information about a new device that was assigned to a Device Enrollment Program instance. The `assetTag` key is provided by Apple's Device Enrollment service -- it's not the same as the device's "Asset Tag" field in Jamf Pro inventory information.


```
{
    "event": {
        "assetTag": "1664194",
        "description": "Mac Pro",
        "deviceAssignedDate": 1552478234000,
        "deviceEnrollmentProgramInstanceId": 1,
        "model": "Mac Pro",
        "serialNumber": "92D8014694C4BE96B3"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 1,
        "name": "Webhook Documentation",
        "webhookEvent": "DeviceAddedToDEP"
```

```
    }
}
```


JSSShutdown & JSSStartup

These events are triggered when the Tomcat service hosting Jamf Pro is stopped or started.


```
{
    "event": {
        "hostAddress": "172.31.16.70",
        "institution": "Company Name",
        "isClusterMaster": false,
        "jssUrl": "https://company.jamfcloud.com/",
        "webApplicationPath": "/usr/local/jss/tomcat/webapps/ROOT"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "name": "Webhook Documentation",
        "webhookEvent": "JSSShutdown"
    }
}
```


Mobile Device Events Overview

The MobileDeviceCommandCompleted, MobileDeviceEnrolled, MobileDevicePushSent and MobileDeviceUnEnrolled events return data in the format depicted below, with their respective values for the `webhookEvent` attribute.


```
{
    "event": {
        "bluetoothMacAddress": "C0:F2:FB:37:04:2B",
        "deviceName": "iPad",
        "icciID": "",
        "imei": "",
        "jssID": 2,
        "model": "iPad4,7",
        "modelDisplay": "iPad mini 3 (Wi-Fi)",
        "osBuild": "14D27",
        "osVersion": "10.2.1",
        "product": null,
        "room": "221",
        "serialNumber": "DLXN69VAG5X8",
        "udid": "270aae10800b6e61a2ee2bbc285eb967050b5994",
        "userDirectoryID": "-1",
        "username": "John Smith",
        "version": "10.2.1",
        "wifiMacAddress": "C0:F2:FB:37:04:1F"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "name": "Webhook Documentation",
        "webhookEvent": "MobileDeviceCommandCompleted"
    }
}
```


MobileDeviceCommandCompleted

This event is triggered when Jamf Pro receives acknowledgement of a completed command for a mobile device. This event will trigger for both successfully completed and failed commands.

MobileDeviceEnrolled

This event is triggered when a mobile device is enrolled or re-enrolled into Jamf Pro.

MobileDevicePushSent

This event is triggered when Jamf Pro issues a push command to a mobile device. Although similar to MobileDeviceCommandCompleted, this event triggers when Jamf Pro sends the command, not after the command has been completed.

MobileDeviceUnEnrolled

This event is triggered when the MDM Profile is manually removed from an enrolled device and when the Unmanage Device remote command is sent from Jamf Pro.

MobileDeviceCheckIn

This event is triggered when a managed mobile device submits inventory.


```
{
    "event": {
        "bluetoothMacAddress": "C0:F2:FB:37:04:2B",
        "deviceName": "iPad",
        "icciID": "",
        "imei": "",
        "ipAddress": "10.15.76.30",
        "jssID": 2,
        "model": "iPad4,7",
        "modelDisplay": "iPad mini 3 (Wi-Fi)",
        "osBuild": "14D27",
        "osVersion": "10.2.1",
        "product": null,
        "room": "221",
        "serialNumber": "DLXN69VAG5X8",
        "udid": "270aae10800b6e61a2ee2bbc285eb967050b5994",
        "userDirectoryID": "-1",
        "username": "John Smith",
        "version": "10.2.1",
        "wifiMacAddress": "C0:F2:FB:37:04:1F"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "name": "Webhook Documentation",
        "webhookEvent": "MobileDeviceCommandCompleted"
    }
}
```


PatchSoftwareTitleUpdated

This event is triggered when Jamf Pro receives an update to a patch title it is subscribed to. For more information on patch software titles, see the Administrator's Guide.


```
{
    "event": {
        "jssID": 1,
        "lastUpdate": 1506031211000,
        "latestVersion": "61.0.3163.100",
        "name": "Google Chrome",
```

```
        "reportUrl": "https://company.jamfcloud.com//view/patch/1/report"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "name": "Webhook Documentation",
        "webhookEvent": "PatchSoftwareTitleUpdated"
    }
}
```


PushSent

This event is triggered when Jamf Pro sends a remote command to either a computer or mobile device.


```
{
    "event": {
        "jssid": 13,
        "type": "PushSent"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "name": "Webhook Documentation",
        "webhookEvent": "PushSent"
    }
}
```


RestAPIOperation

This event is triggered whenever any authorized HTTP request is made to a supported resource for the Classic API. Unauthorized requests, failed requests and non-existent resources will not trigger this event.


```
{
    "event": {
        "authorizedUsername": "administrator",
        "objectID": 34,
        "objectName": "Self Service Mobile",
        "objectTypeName": "Mobile Device Application",
        "operationSuccessful": true,
        "restAPIOperationType": "GET"
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 7,
        "name": "Webhook Documentation",
        "webhookEvent": "RestAPIOperation"
    }
}
```


SmartGroupComputerMembershipChange and **SmartGroupMobileDeviceMembershipChange**

This event is triggered whenever a managed computer or mobile device falls-into or out of membership of a smart group. This can happen when a device reports in with updated inventory information or when the Jamf Pro administrator changes the criteria of the smart group. The webhook body will include the name of the smart group and a list of any devices that have fallen in or out of membership. The devices are listed within the `groupAddedDevices` or `groupRemovedDevices` arrays. You can select a specific Smart Group to trigger each webhook, which is useful for our purposes; there are probably a lot of smart groups in Jamf Pro that you won't be interested in monitoring in Splunk.

By default, the list of devices falling in or out of membership includes only the Jamf Pro device ID of the devices. If you're already collecting Jamf Pro information in Splunk using the Jamf Pro API Add-on for Splunk, you could use a join operation in your Splunk search to link in additional information about the device, like the user, model, etc. If you're not pulling device data into Splunk, you also have the option of adding additional fields beyond just device ID to the webhook data payload. This functionality is configured via the Classic API. For more information see the POST/PUT operations of the `/webhooks` endpoint in the Classic API documentation.

This example was sent by a mobile device smart group membership change:


```
{
    "event": {
        "computer": false,
        "groupAddedDevices": [],
        "groupAddedDevicesIds":
        [1,2,3,4,5],
        "groupRemovedDevices": [],
        "groupRemovedDevicesIds":
        [6,7,8,9,10],
        "jssid": 8,
        "name": "Smart Group Name",
        "smartGroup": true
    },
    "webhook": {
        "eventTimestamp": 1553550275590,
        "id": 1,
        "name": "Webhook Documentation",
        "webhookEvent": "SmartGroupMobileDeviceMembershipChange"
    }
}
```

Appendix 6: Jamf Protect Event Data Structure

An example of the JSON sent by a Jamf Protect log notification is shown here, noting the information it contains and some of the fields that might be of interest for analysis.

```
{
  "input": {
    "match": {
      "facts": [
        {
          "actions": [
            {
              "name": "Log",
              "parameters": {}
            }
          ],
          "human": "Created an application helper login item",
          "context": [
            {
              "name": "ItemBinary",
              "value": "/Applications/Thunder.app/Contents/Library/LoginItems/ThunderHelper.app/
Contents/MacOS/ThunderHelper", [The executable that was set to run automatically on login]
              "valueType": "Binary"
            },
            {
              "name": "ItemName",
              "value": "ThunderHelper",
              "valueType": "String"
            }
          ],
          "uuid": "97A98D39-8003-4D34-9C78-77DB554A7C2F",
          "version": null,
          "tags": [
            "Persistence",
            "MITREattack",
            "LoginItems"
          ],
          "name": "AppLoginItem" [The fact's name indicates what happened to trigger the report.]
        }
      ],
      "tags": [.                  [Tags are set on detections for easy grouping.]
        "LoginItems",
        "Persistence",
        "MITREattack"
      ],
      "uuid": "68852C34-6EAC-47B0-8B37-94F6DA76E12A",
      "event": {
        "pid": 8497,    [Process ID is useful when obtaining details for a process.]
        "uid": 502,
        "gid": 80,
        "eventID": 16779243857974,
        "dev": 16777220,
        "uuid": "7AA2C9EE-881F-45B3-B70E-04F81F60EE2D",
        "path": "/Applications/Thunder.app/Contents/Library/LoginItems/ThunderHelper.app",
        "timestamp": 1581242041.636719,
        "iNode": 81060,
        "type": 7
      },
      "actions": [              [What actions is this detection set to trigger?]
        {
          "name": "Log",
          "parameters": {}
        }
      ],
      "context": [
        {
          "name": "ItemName",
          "value": "ThunderHelper",
          "valueType": "String"
```

```
        },
        {
          "name": "ItemBinary",
          "value": "/Applications/Thunder.app/Contents/Library/LoginItems/ThunderHelper.app/
Contents/MacOS/ThunderHelper",
          "valueType": "Binary"
        }
      ]
    },
    "reportType": "log", [The report type is either "log" or "event"]
    "eventType": "GPFSEvent",
    "host": {
      "ips": [
        "172.20.6.52" [IP address(es) of the Mac computer can be used to join network logs]
      ],
      "hostname": "Jeff\u7684MacBook Air", [The Mac's computer name]
      "serial": "FVFX" [The Mac's Serial Number. We can use this to join data from Jamf Pro]
    },
    "related": {
      "users": [ [What accounts are on the device?]
        {
          "uid": 0,
          "name": "root",
          "uuid": "FVFX"
        },
        {
          "uid": 502,
          "name": "jeffxu",
          "uuid": "FVFX"
        }
      ],
      "files": [. [Signing details, entitlements, and extended attributes for the executable
                   This is useful.
                   For example, if something of concern is found, what other items have
                    been logged that were signed by the same developer or team?]
        {
          "modified": 1578883128,
          "downloadedFrom": null,
          "changed": 1581242041,
          "signingInfo": {
            "status": 0,
            "authorities": [
              "Developer ID Application: Xunlei Computer LTD (SXKSDVK696)",
              "Developer ID Certification Authority",
              "Apple Root CA"
            ],
            "teamid": "SXKSDVK696",
            "signerType": 2,
            "statusMessage": "No error.",
            "entitlements": [],
            "appid": "com.xunlei.Thunder.ThunderHelper"
          },
          "signatureCalculated": true,
          "xattrs": [
            "com.apple.quarantine"
          ],
          "path": "/Applications/Thunder.app/Contents/Library/LoginItems/ThunderHelper.app",
          "inode": 81060,
          "sha1hex": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
          "size": 96,
          "isDirectory": true,
          "sha256hex": "e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
          "uid": 502,
          "isDownload": false,
          "gid": 80,
          "fsid": 16777220,
          "isScreenShot": false,
          "mode": 16877,
          "isAppBundle": true,
          "created": 1578883128,
          "accessed": 1581242041
        }
```

```
    ],
    "binaries": [ [What executable stack is behind the detection?]

      {
        "modified": 1578883131,
        "downloadedFrom": null,
        "changed": 1581242041,
        "signingInfo": {
          "status": 0,
          "authorities": [
            "Developer ID Application: Xunlei Computer LTD (SXKSDVK696)",
            "Developer ID Certification Authority",
            "Apple Root CA"
          ],
          "teamid": "SXKSDVK696",
          "signerType": 2,
          "statusMessage": "No error.",
          "entitlements": [],
          "appid": "com.xunlei.Thunder.ThunderHelper"
        },
        "signatureCalculated": true,
        "xattrs": [
          "com.apple.quarantine"
        ],
        "path": "/Applications/Thunder.app/Contents/Library/LoginItems/ThunderHelper.app/
Contents/MacOS/ThunderHelper",
        "inode": 81067,
        "sha1hex": "5b00c2e81aa2d14e6e109ba1f4ac784615647c30",
        "size": 486176,
        "isDirectory": false,
        "sha256hex": "890478837e18c352c5a83d3e6b998c6f08308e0711afa91786fce14ffd34f3d4",
        "uid": 502,
        "isDownload": false,
        "gid": 80,
        "fsid": 16777220,
        "isScreenShot": false,
        "mode": 33261,
        "isAppBundle": false,
        "created": 1578883131,
        "accessed": 1581242041
      },
      {
        "modified": 1579784288,
        "downloadedFrom": null,
        "changed": 1580930437,
        "signingInfo": {
          "status": 0,
          "authorities": [
            "Software Signing",
            "Apple Code Signing Certification Authority",
            "Apple Root CA"
          ],
          "teamid": "",
          "signerType": 0,
          "statusMessage": "No error.",
          "entitlements": [],
          "appid": "com.apple.DesktopServicesHelper"
        },
        "signatureCalculated": true,
        "xattrs": [],
        "path": "/System/Library/PrivateFrameworks/DesktopServicesPriv.framework/Versions/A/
Resources/DesktopServicesHelper",
        "inode": 1152921500312195898,
        "sha1hex": "c07b2841941c6a38b633b5930d45eee189921fc9",
        "size": 531872,
        "isDirectory": false,
        "sha256hex": "d943c6abc097114d58e4b6abd8e0b68d56970e4b455a83a5fdd8f837f3ece236",
        "uid": 0,
        "isDownload": false,
        "gid": 0,
        "fsid": 16777220,
        "isScreenShot": false,
```

```
          "mode": 33261,
          "isAppBundle": false,
          "created": 1579784288,
          "accessed": 1579784288
        }
      ],
      "groups": [    [User groups of the process user. Are they running as an admin?]

        {
          "gid": 80,
          "name": "admin",    [Yes, they are.]
          "uuid": "FVFXM8G4JK7750"
        },
        {
          "gid": 0,
          "name": "wheel",
          "uuid": "FVFXM8G4JK770"
        }
      ],
      "processes": [  [What process stack is behind the detection?]
        {
          "uuid": "EF1F08EA-41E8-4760-8416-2D2FFF947B02",
          "ruid": 0,
          "uid": 0,
          "endTimestamp": 1581242042,
          "exitCode": 0,
          "startTimestamp": 1581242040,
          "ppid": 1,
          "appPath": null,
          "path": "/System/Library/PrivateFrameworks/DesktopServicesPriv.framework/Versions/A/
Resources/DesktopServicesHelper",
          "gid": 0,
          "rgid": 0,
          "args": [
            "/System/Library/PrivateFrameworks/DesktopServicesPriv.framework/Resources/
DesktopServicesHelper"
          ],
          "signingInfo": {
            "status": 0,
            "authorities": [
              "Software Signing",
              "Apple Code Signing Certification Authority",
              "Apple Root CA"
            ],
            "teamid": "",
            "signerType": 0,
            "statusMessage": "No error.",
            "entitlements": [],
            "appid": "com.apple.DesktopServicesHelper"
          },
          "pid": 8497,
          "pgid": 8497,
          "name": "DesktopServicesHelper"
        }
      ]
    }
  },
  "caid": "b430a234c148552610b60686ea7088200a197378e26cf9cba6d867bf9606ae53",
  "certid": "3aa90070b9e49a09a2855b30bac5f225d97dcd8a087ea237686d1598fcb68ddc"
}
```

Appendix 7: Mitre ATT&CK Matrix for macOS

What should we be looking for when developing a security monitoring system? The The Mitre ATT&CK framework suggest some common adversary tactics and techniques based on real-world observations.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Dylib Hijacking	Binary Padding	Bash History	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Account Access Removal
Exploit Public-Facing Application	Command-Line Interface	Browser Extensions	Elevated Execution with Prompt	Clear Command History	Brute Force	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Destruction
Hardware Additions	Exploitation for Client Execution	Create Account	Emond	Code Signing	Credential Dumping	Browser Bookmark Discovery	Exploitation of Remote Services	Clipboard Data	Connection Proxy	Data Encrypted	Data Encrypted for Impact
Spearphishing Attachment	Graphical User Interface	Dylib Hijacking	Exploitation for Privilege Escalation	Compile After Delivery	Credentials from Web Browsers	File and Directory Discovery	Internal Spearphishing	Data from Information Repositories	Custom Command and Control Protocol	Data Transfer Size Limits	Defacement
Spearphishing Link	Launchctl	Emond	Launch Daemon	Connection Proxy	Credentials in Files	Network Service Scanning	Logon Scripts	Data from Local System	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Content Wipe
Spearphishing via Service	Local Job Scheduling	Hidden Files and Directories	Plist Modification	Disabling Security Tools	Exploitation for Credential Access	Network Share Discovery	Remote File Copy	Data from Network Shared Drive	Data Encoding	Exfiltration Over Command and Control Channel	Disk Structure Wipe
Supply Chain Compromise	Scripting	Kernel Modules and Extensions	Process Injection	Execution Guardrails	Input Capture	Network Sniffing	Remote Services	Data from Removable Media	Data Obfuscation	Exfiltration Over Other Network Medium	Endpoint Denial of Service
Trusted Relationship	Source	Launch Agent	Setuid and Setgid	Exploitation for Defense Evasion	Input Prompt	Password Policy Discovery	SSH Hijacking	Data Staged	Domain Fronting	Exfiltration Over Physical Medium	Firmware Corruption
Valid Accounts	Space after Filename	Launch Daemon	Startup Items	File and Directory Permissions Modification	Keychain	Peripheral Device Discovery	Third-party Software	Input Capture	Domain Generation Algorithms	Scheduled Transfer	Inhibit System Recovery
	Third-party Software	Launchctl	Sudo	File Deletion	Network Sniffing	Permission Groups Discovery		Screen Capture	Fallback Channels		Network Denial of Service
	Trap	LC_LOAD_DYLIB Addition	Sudo Caching	Gatekeeper Bypass	Private Keys	Process Discovery		Video Capture	Multi-hop Proxy		Resource Hijacking
	User Execution	Local Job Scheduling	Valid Accounts	Hidden Files and Directories	Securityd Memory	Remote System Discovery			Multi-Stage Channels		Runtime Data Manipulation
		Login Item	Web Shell	Hidden Users	Steal Web Session Cookie	Security Software Discovery			Multiband Communication		Stored Data Manipulation
		Logon Scripts		Hidden Window	Two-Factor Authentication Interception	Software Discovery			Multilayer Encryption		System Shutdown/ Reboot
		Plist Modification		HISTCONTROL		System Information Discovery			Port Knocking		Transmitted Data Manipulation
		Port Knocking		Indicator Removal from Tools		System Network Configuration Discovery			Remote Access Tools		
		Rc.common		Indicator Removal on Host		System Network Connections Discovery			Remote File Copy		

Re-opened Applications	Install Root Certificate	System Owner/User Discovery	Standard Application Layer Protocol
Redundant Access	Launchctl	Virtualization/ Sandbox Evasion	Standard Cryptographic Protocol
Setuid and Setgid	LC_MAIN Hijacking		Standard Non-Application Layer Protocol
Startup Items	Masquerading		Uncommonly Used Port
Trap	Obfuscated Files or Information		Web Service
Valid Accounts	Plist Modification		
Web Shell	Port Knocking		
	Process Injection		
	Redundant Access		
	Rootkit		
	Scripting		
	Software Packing		
	Space after Filename		
	Timestomp		
	Valid Accounts		
	Virtualization/ Sandbox Evasion		
	Web Service		