splunk>

Creating Splunk 6.4 Knowledge Objects Class Lab Exercises

Lab typographical conventions

`{student ID}` indicates you should replace this with your student number.

`{server-name}` indicates you should substitute the server name assigned to this class.

There are a number of source types used in these lab exercises. The lab instructions refer to these source types by the types of data they represent:

Type	Sourcetype	Interesting Fields
AD/DNS	winauthentication_security (corporate network)	bcg_ip, bcg_workstation, fname, lname, location, rfid, splunk_role
	WinEventLog:Security (engineering network)	Account_Domain, Account_Name, action, app, Authentication_Package, Type, User
BI server	sales_entries	AcctCode, CustomerID, TransactionID
Email data	cisco_esa	dcid, icid, mailfrom, mailto, mid
Web appliance data	cisco_wsa_squid	action, bandwidth, cs_method, cs_mime_type, cs_url, cs_username, sc_bytes, sc_http_status, sc_result_code, severity, src_ip, status, url, usage, x_mcafee_virus_name, x_wbrs_score, x_webcat_code_abbr
Online transactions	access_combined	action, bytes, categoryId, clientip, itemId, JSESSIONID, price, productId, product_name, referer, referer_domain, sale_price, status, user, useragent
Retail sales	vendor_sales	AcctID, categoryId, product_name, productId, sale_price, Vendor, VendorCity, VendorCountry, VendorID, VendorStateProvince
Web server	linux_secure	action, app, COMMAND, dest, process, src_city, src_country, src_ip, src_port, user, vendor_action
Windows server logs	win_audit	

****For all exercises, keep the permissions for your knowledge objects private.****

Module 3 Lab Exercise: Creating Lookups

Description

In this lab exercise, you create a new automatic lookup that provides additional information for the vendors selling Buttercup Games products. ****Note: this automatic lookup is required for Lab Exercise 9.**

Tasks

Task 1: Log into Splunk on the classroom server (server name provided by instructor). After you log in, review the data sources using the Data Summary page.

Task 2: Change your account name and time zone setting to reflect your local time.

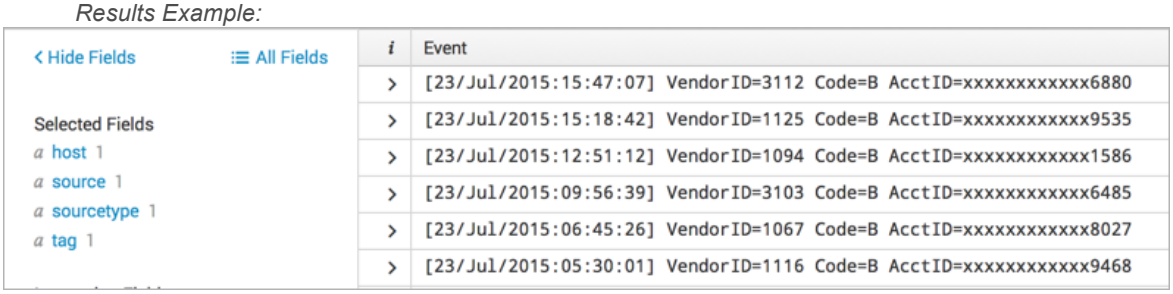
Scenario: The `vendor_sales` source type does not contain vendor locations. Reports need to be created to show how game sales are performing based on region, country, state, and city. A lookup is needed to provide this information when searches are performed.

Task 3: Search retail sales for Dream Crusher sales over the **last 30 days** and save your search as **{student name}_DreamCrusherSales.**

sourcetype=vendor_sales product_name="Dream Crusher"

As you can see, the raw data has a limited amount of useful and detailed information. For example, the vendor name or city is not listed.

Results Example:



Task 4: Add the **vendor_lookup.csv** lookup file in the **search** app with a destination file name of **vendor_lookup.csv**.

Task 5: Create a file-based lookup definition called **vendor_lookup** in the **search** app using the **vendor_lookup.csv** lookup file.

Task 6: Verify the lookup data using the `inputlookup` search command.

| inputlookup vendor_lookup

Results Example:

Vendor	VendorCity	VendorCountry	VendorID	VendorStateProvince
Anchorage Gaming	Anchorage	United States	1001	Alaska
Games of Salt Lake	Salt Lake City	United States	1002	Utah
New Jack Games	New York	United States	1003	New York
Seals Gaming	San Francisco	United States	1004	California
Lost Angels Games	Los Angeles	United States	1005	California
Flyin' Hawaiian Hobbyist	Honolulu	United States	1006	Hawaii
Flyin' Hawaiian Hobbyist	Kahului	United States	1007	Hawaii
Phoenix Games	Phoenix	United States	1008	Arizona
Mile High Games	Denver	United States	1009	Colorado
Beantown Games	Boston	United States	1010	Massachusetts
Seattle Games	Seattle	United States	1011	Washington

Task 7: Use your lookup in a search. Search the vendor data for all Dream Crusher game sales worldwide for the **last 30 days**.

sourcetype=vendor_sales product_name="Dream Crusher" | lookup vendor_lookup VendorID OUTPUT VendorCountry |stats sum(price) as sales by VendorCountry

Results Example:

VendorCountry	sales
United States	9997.50
India	479.88
Canada	439.89
China (PRC)	159.96
Austria	79.98
Bolivia	79.98
Brazil	79.98
Chile	79.98
Ecuador	79.98
Egypt	79.98
Estonia	79.98
Ethiopia	79.98

Task 8: Create an automatic lookup definition called **vendor_auto_lookup** in the **search** app. Use the **vendor_lookup** table and apply it to the sourcetype `vendor_sales`. Use **VendorID** as the lookup input field and **Vendor**, **VendorCity**, **VendorStateProvince**, and **VendorCountry** as the lookup output fields.

Task 9: Verify your automatic lookup is working. Search the `vendor_sales` source type for the total amount of Manganiello Bros. games sold by country in the **last 30 days**. Sort you results in descending order.

sourcetype=vendor_sales product_name="Manganiello Bros." | stats count, sum(price) as sales by VendorCountry | sort -sales

Now, you will notice that the Vendor, VendorCity, VendorStateProvince, and VendorCountry fields appear in the fields sidebar when you perform a search on vendor_sales data.

Results Example:

VendorCountry	count	sales
United States	945	77055.30
Canada	91	7420.14
Germany	32	2609.28
Italy	29	2364.66
China (PRC)	28	2283.12
India	25	2038.50
France	24	1956.96
United Kingdom	23	1875.42
Spain	19	1549.26
Brazil	18	1467.72
Egypt	15	1223.10
Israel	15	1223.10
Japan	14	1141.56
Poland	14	1141.56
Australia	11	896.94
Belgium	10	815.40
Denmark	10	815.40
Hungary	10	815.40
Ireland	10	815.40
South Africa	9	733.86

Module 4 Lab Exercise: Working with Field Aliases and Calculated Fields

Description

This lab exercise walks you through the process of creating field aliases and calculated fields.

Tasks

Scenario: The IT Ops team runs reports for all employee access but the user name field is not consistent across the different source types.

Task 1: Perform a search on the `cisco_wsa_squid` source type for the **last 24 hours**. Note that the user information is defined in the cs_username field. Create a field alias in the **search** app called **cisco_wsa_squid_aliases**. Apply the field alias to the **cisco_wsa_squid** sourcetype and create the field alias **user** for the **cs_username** field.

Re-run your search and examine the user field and values.

Results Example:

Perform a search on the `cisco_firewall` source type for the **last 24 hours**. Note that the user information is defined in the Username field. Create a field alias in the **search** app called **cisco_firewall_aliases**. Apply the field alias to the **cisco_firewall** sourcetype and create the field alias **user** for the **Username** field. Perform the following search**:** **`sourcetype=cisco* user=*`**

Do you receive results from the `cisco_wsa_squid` and `cisco_firewall` sourcetypes?

Scenario: The IT Ops team is monitoring bandwidth usage for all users for the last 30 days, but the data is reported in bytes. The team needs the usage to be measured in megabytes.

Task 2: Create a calculated field called **bandwidth** that converts bytes to MB in the **search** app. Apply the calculated field to the **`cisco_wsa_squid`** sourcetype with an eval expression of `sc_bytes/(1024*1024)`

To verify your work perform a search on the `cisco_wsa_squid` source type that shows the total bandwidth by usage.

sourcetype=cisco_w* | stats sum(bandwidth) as "Bandwidth (MB)" by usage

Results Example:

usage	Bandwidth (MB)
Borderline	9.133331
Business	14.596395
Personal	78.236745
Unknown	20.043213
Violation	1.063354

Supplemental Exercise:

Scenario: The IT Ops team wants to correlate data from multiple source types using the `src` and `http_method` fields. However, these fields are called `clientip` and `method` in the `access_combined` source type.

Task: Create field aliases for `access_combined` so `src` and `http_method` can be used in searches.

Module 5 Lab Exercise: Creating Field Extractions

Description

This lab exercise walks you through the process of creating a regex and delimiters field extractions. ****Note: this field extraction is required for Lab Exercise 7.**

Steps

Scenario: Access to the Linux server also needs to be monitored for events in the last 24 hours. However, the IP address and port number are not automatically extracted.

Task 1: Use the FX to extract the IP address and port fields using the Regular Expression method. Extract the IP address field as **src_ip** and the port field as **port.**

To verify your work, search for events in the `linux_secure` source type in the **last 24 hours**. List the top ports by IP address.

Note: Notice that **:: is extracted as a src_ip value. Type the following in the filter field: **src_ip!=::**

sourcetype=linux_secure | top port by src_ip

Results Example:

src_ip	port	count	percent
10.1.10.172	4717	3	0.273224
10.1.10.172	3567	3	0.273224
10.1.10.172	2558	3	0.273224
10.1.10.172	2080	3	0.273224
10.1.10.172	1713	3	0.273224
10.1.10.172	1676	3	0.273224
10.2.10.163	4673	3	0.303644
10.2.10.163	4541	3	0.303644
10.2.10.163	1063	3	0.303644
10.2.10.163	4884	2	0.202429
10.2.10.163	4824	2	0.202429

Scenario: The `win_audit` source type has been added to the Splunk environment and IT Ops needs to monitor events in the last 24 hours. However, the log file is in csv format, doesn't contain headers, and none of the fields are extracted.

Task 2: Search the `win_audit` sourcetype for **the last 24 hours**. Take note of the event format and fields. Use the FX to extract the fields using the delimiters method. Rename the fields using the following values:

field1:	**Time**
field2:	**EventCode**
field3:	**EventType**
field4:	**Type**
field5:	**ComputerName**
field6:	**LogName**
field7:	**RecordNumber**

Save your extraction as **sysmon**.

Module 6 Lab Exercise: Creating Tags and Event Types

Description

This lab exercise walks you through the process of creating tags and event types.

Tasks

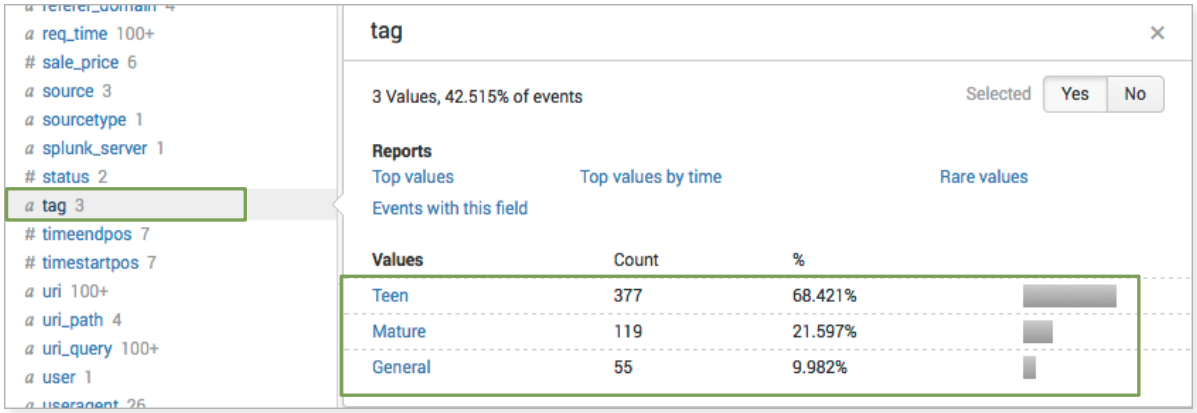
Scenario: The SVP of Marketing wants to easily identify products by a rating system that is not currently tracked in the data. The ratings of General, Teen, and Mature need to be applied to the games within the different categories. Reports will be run for the last 24 hours.

Task 1: Create tags using the `categoryId` field to identify product ratings for data in `access_combined`. Tag sports games as **General**, strategy games as **Teen**, and shooter games as **Mature**.

Task 2: Perform a search and verify the tags are created. Modify the search to limit results to only game categories tagged as **Teen**

Hint: `tag=Teen`. Also note that tags are case sensitive. A search for `tag=teen` produces no results.

Results Example:



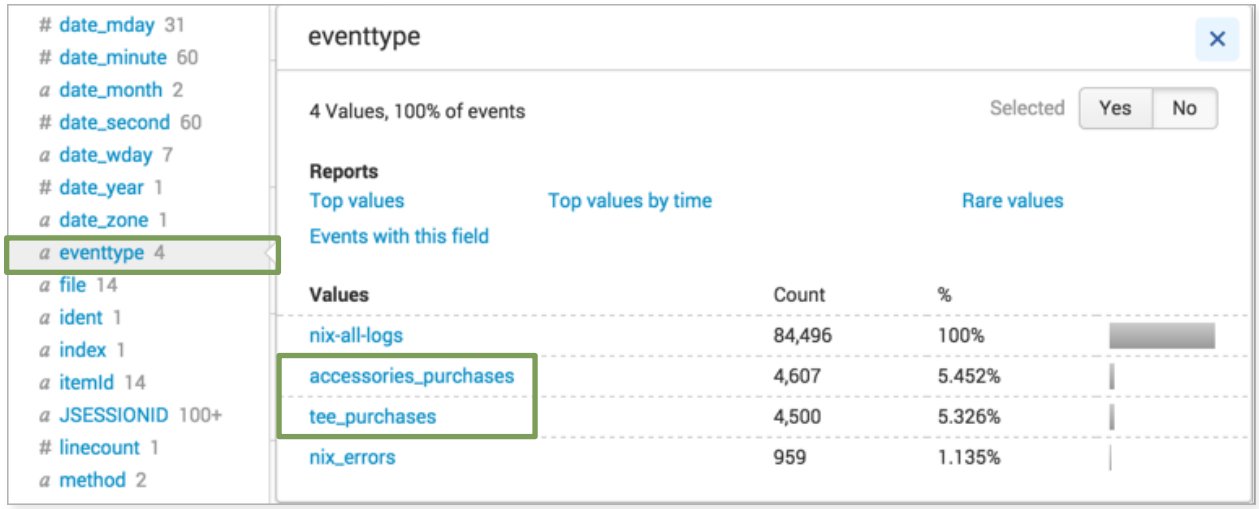
Scenario: The Sales team wants to track online sales for the last 24 hours. However, they want to easily identify purchases that are categorized by item.

Task 3: Use the search app to create an event type for accessories called **accessories_purchases** and an event type for tees called **tee_purchases**.

Hint: To create the first event type: `sourcetype=access_combined action=purchase categoryId=accessories`

To verify your event types were created, perform a search for purchase events with `categoryId` values.

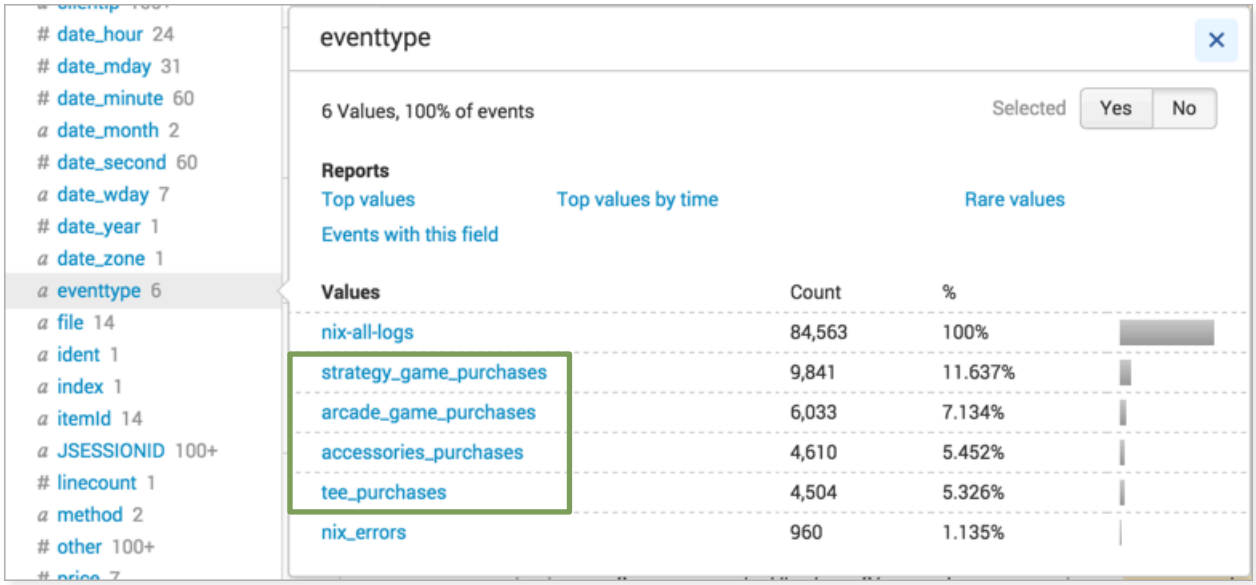
Results Example:



Task 4: Use the Event Type Settings page to create an event type for strategy games called **strategy_game_purchases** and an event type for arcade games called **arcade_game_purchases**.

To verify your work, return to the **Search & Reporting** app and run a search to verify that your event types are being returned.

Results Example:



Note: Based on add-ons or apps you have installed, additional event types may be displayed. In this example, nix-all-logs is added by the *NIX app.

Supplemental Exercise:

Task: Tag these event types as purchases. Perform a search for the **purchases** tag. What types of events do you receive?

Module 7 Lab Exercise: Creating and Using Workflow Actions

Description

In this exercise, you learn to create GET and Search workflow actions. You will use fields from the field extraction lab exercise.

****Note**: You must have successfully completed Lab Exercise 5 to complete this lab exercise.

Tasks

Scenario: Hackers are continually trying to log into the Linux server. IT Ops analysts need to track ongoing attempts by external sources trying to log in with invalid credentials.

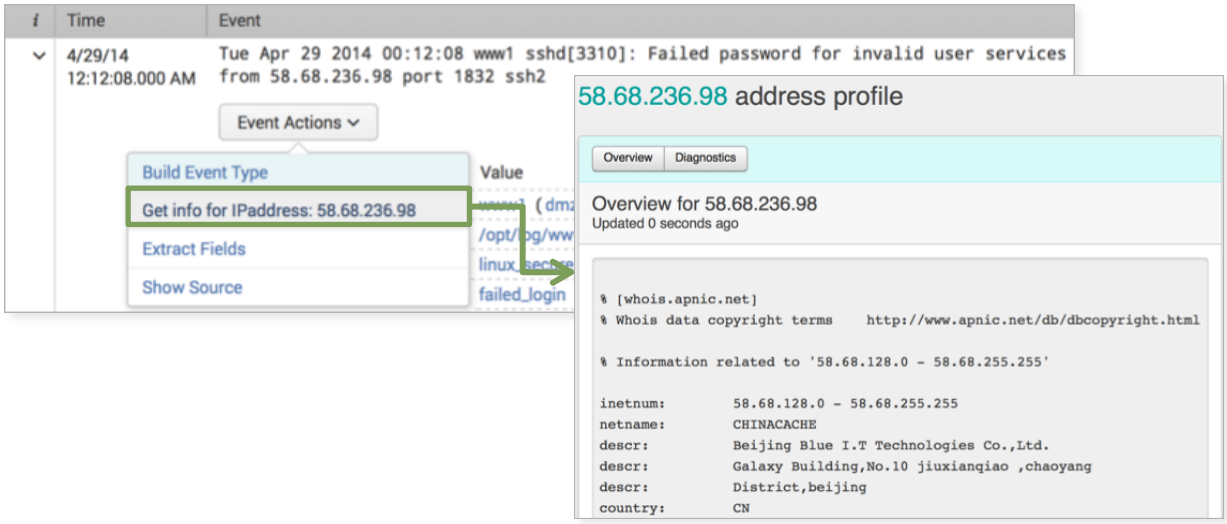
Task 1: Create a **GET** workflow action called **get_whois_info** in the **search** app that opens a new browser window with information about the source IP address. Create the label **Get info for IPaddress: src_ip** and the workflow action should only apply to the **src_ip** field. The action type is **link** and when the URI **http://who.is/whois-ip/ip-address/src_ip** is launched, open it in a **new window** as a **get** link method.

****Note:** If who.is is not behaving as expected, try http://whois.domaintools.com/src_ip.

Verify your workflow action works as expected. Return to the **Search & Reporting** app and search for `sourcetype=linux_secure src_ip=*` over the **last 24 hours**.

HINT: Click the search menu option to refresh your browser.

Results Example:

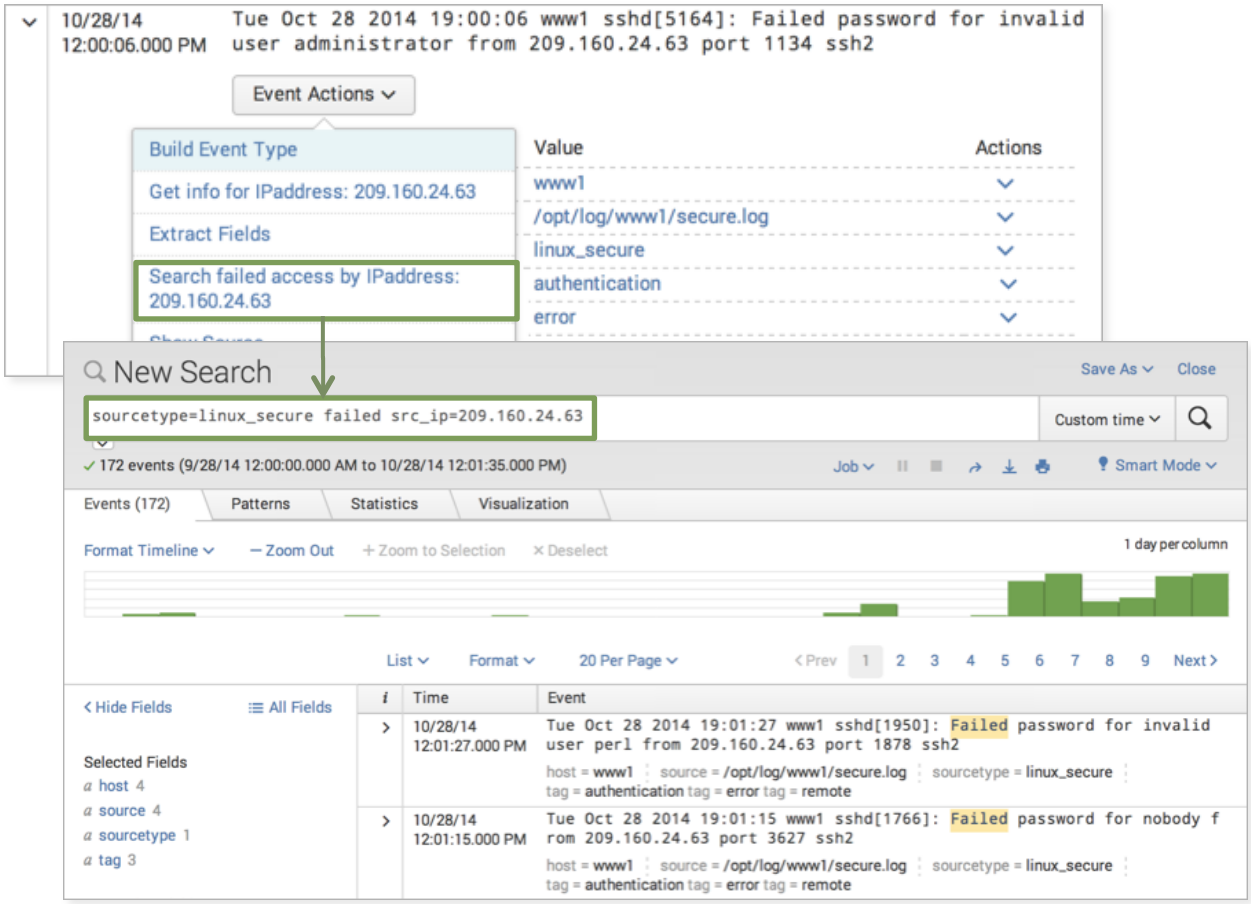


Task 2: Create a search workflow action in the **search** app called **search_access_by_ipaddress** that performs a search for all failed password events associated with a specific IP address. Create the label **Search failed access by IPaddress: src_ip** and should only apply to the **src_ip** field**.** The action type is **search** and when the search **`sourcetype=linux_secure failed src_ip=$src_ip$`** is run, open the search in a **new window** using **the same time range as the search that created the field listing.**

HINT: Click the search menu option to refresh your browser.

Verify your workflow action works as expected. Return to the **Search & Reporting** app and search for `sourcetype=linux_secure src_ip=*` over the **last 24 hours**.

Results Example:



Module 8 Lab Exercise: Creating Alerts

Description

This lab walks you through the process to create an alert.

Tasks

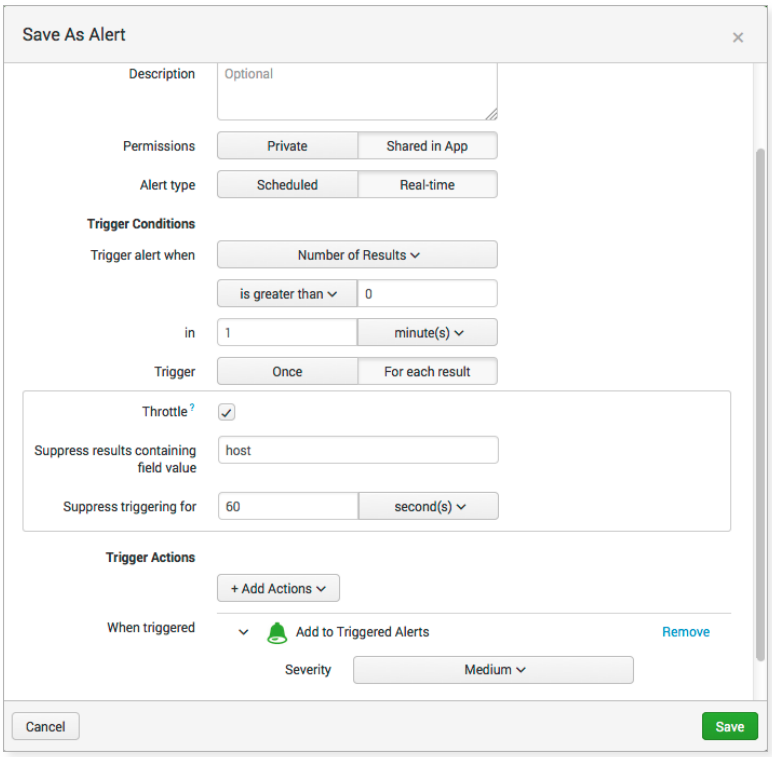
Scenario: **For security reasons, you need to monitor failed login attempts into the web servers. You are only interested in failed logins from known user accounts. You need to track these because they can be more dangerous than unknown users. To gain access, attackers need a user name and a password. You want to be notified when there is more than one failed login attempt within one minute.**

Task 1: Create a search to identify specific types of failed logins. Search for failed and invalid logins to the `linux_secure` source type in the **last 60 minutes**.

sourcetype=linux_secure failed password NOT invalid

Task 2: Create a **real-time alert** called **<your name> - Login attempts** and set the permissions as **Shared in App**. Trigger the alert when the **number of results is greater than 0** within one minute (this setting is used for testing purposes. Once the alert is verified you can change this value). This alert should trigger **once for each result**. Enable throttling and suppress any results containing the field value **host** for 60 seconds. When the alert is triggered, add it to the **triggered alert** list with a severity set to **high**.

Example:



Review the permissions and view your triggered alerts.

Note: It might take a few minutes for your alert to appear.

Task 3: Disable the alert.

Module 9 Lab Exercise: Creating and Using Macros

Description

This lab exercise walks you through the steps for creating a basic macro and a macro with arguments. You will use fields that were added from the lookup lab.

****Note**: You must have successfully completed Lab Exercise 3 to complete this lab exercise.

Tasks

Scenario: **The VP of Sales wants to run ad-hoc searches to determine how much product is being sold in a given month, in various countries. He also wants to easily convert the sales to US Dollars based on the current exchange rates.**

Task 1: Create a basic macro that lists the monthly total sales in the US.

Name the macro **US_sales** and use the following search string:

```
sourcetype=vendor_sales VendorCountry="United States"  | stats sum(price) as USD by product_name | eval USD = "$" + tostring(USD,"commas")
```

Task 2: Use the macro to search for sales over the **last 30 days**.

Results Example:

product_name	USD
Benign Space Debris	$2,848.86
Curling 2014	$3,518.24
Dream Crusher	$21,514.62
Final Sequel	$8,596.56
Fire Resistance Suit of Provolone	$1,695.75
Holy Blade of Gouda	$1,743.09
Manganiello Bros.	$11,717.07
Manganiello Bros. Tee	$3,066.93
Mediocre Kingdoms	$5,322.87

Task 3: Create a macro with currency, currency symbol, and rate as arguments.

Name the macro: **monthly_sales(3)** and use the following search:

```
| stats sum(price) as USD by product_name |  eval $currency$ = "$symbol$" + tostring(USD*$rate$, "commas") | eval USD = "$" + tostring(USD, "commas")
```

Hint: currency,symbol,rate (order of the arguments must match the order of the values in the search string)

Task 4: Use your macro with arguments in a search where `sourcetype=vendor_sales` where the `VendorCountry` is Germany, France, or Italy. Use the macro and pass the arguments `euro`, `€`, and `.79` for results in the Last 30 days. Copy/paste the `€` symbol from this document.

Hint: `` `macroname(currency,symbol,rate)` ``

sourcetype=vendor_sales VendorCountry=Germany OR VendorCountry=France OR VendorCountry=Italy `monthly_sales(euro,€,.79)`

Run the search again for sales in the UK with the following arguments `GBP`, `£`, and `.64`. Copy/paste the `£` from this document.

sourcetype=vendor_sales VendorCountry="United Kingdom" `monthly_sales(GBP,£,.64)`

Results Example:

product_name	USD	GBP
Benign Space Debris	$174.93	£112
Curling 2014	$219.89	£141
Dream Crusher	$119.97	£77
Final Sequel	$49.98	£32
Fire Resistance Suit of Provolone	$35.91	£23
Holy Blade of Gouda	$41.93	£27
Manganiello Bros.	$319.92	£205
Manganiello Bros. Tee	$109.89	£70
Mediocre Kingdoms	$99.96	£64
Orvil the Wolverine	$399.90	£256
SIM Cubicle	$359.82	£230
World of Cheese	$199.92	£128
World of Cheese Tee	$129.87	£83

Supplemental Exercise:

Task: Edit your macro and use the macro validation fields. Use the **isnum** expression to validate the rate field and provide an appropriate error message. Then test your macro by entering a string value for the rate.

Module 10 Lab Exercise: Creating a Data Model

Description

This lab exercise walks you through the process to create a data model. After the data model is created, create a pivot to verify your data model provides the expected results.

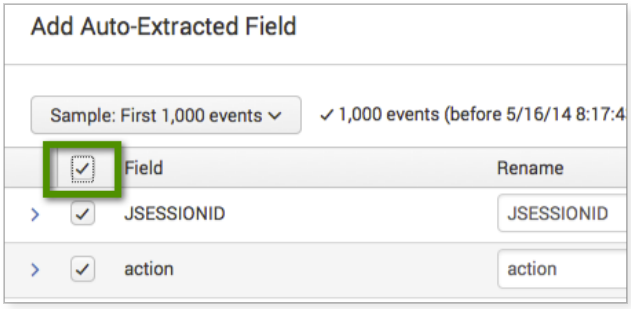
Tasks

Scenario: The VP of Sales wants to run reports based on daily activity from the online store.

Task 1: Create a data model titled **Buttercup Games Site Activity**. Add the **Web Requests** root event with a constraint of `sourcetype=access_combined`. The root event will be the base search for all child events.

Task 2: Add all auto-extracted fields.

Example:



Rename the following fields for pivot users:

action > action taken
bytes > size
categoryId > product category
clientip > client IP
productId > product ID
product_name > product name
req_time > request time

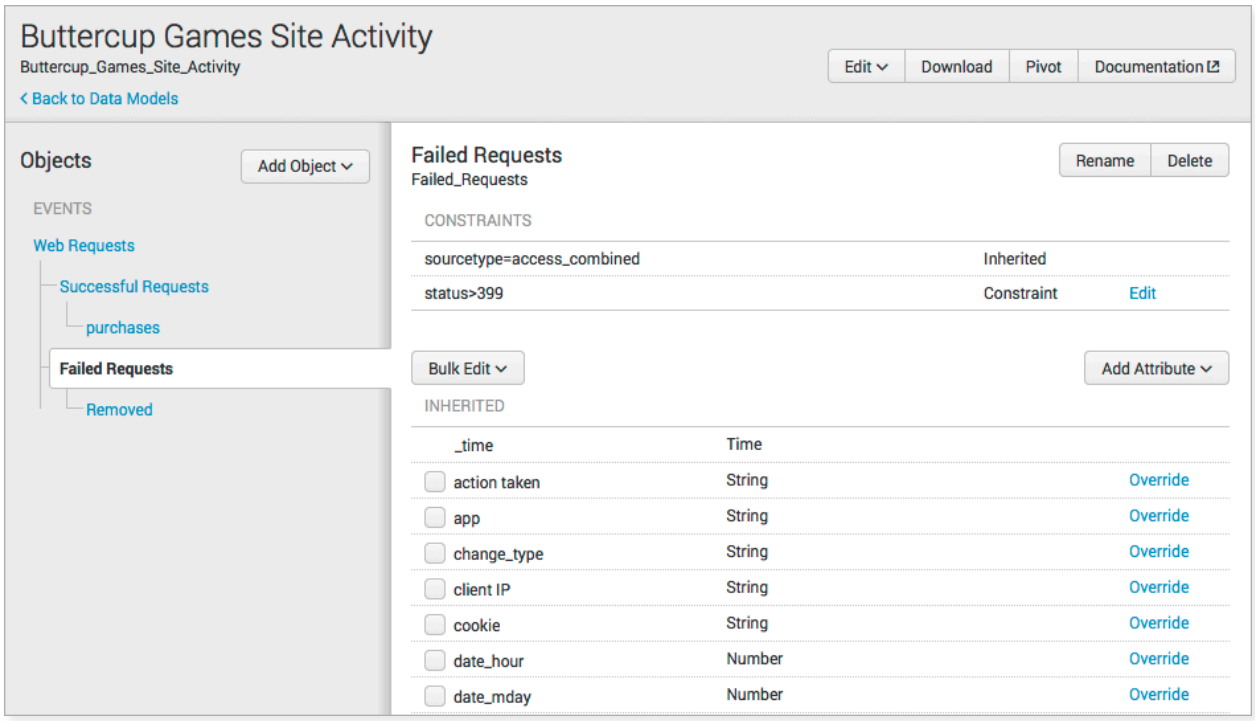
Task 3: Add a child event for actions that are successful. Create an object name called **Successful Requests** with a constraint of **`status<400`**.

Under the Successful Requests object, add a child object called **purchases** with the constraint of **`action=purchase productId=*`**.

Under the Web Requests object, add a child object called **Failed Requests** with the constraint of **`status>399`**.

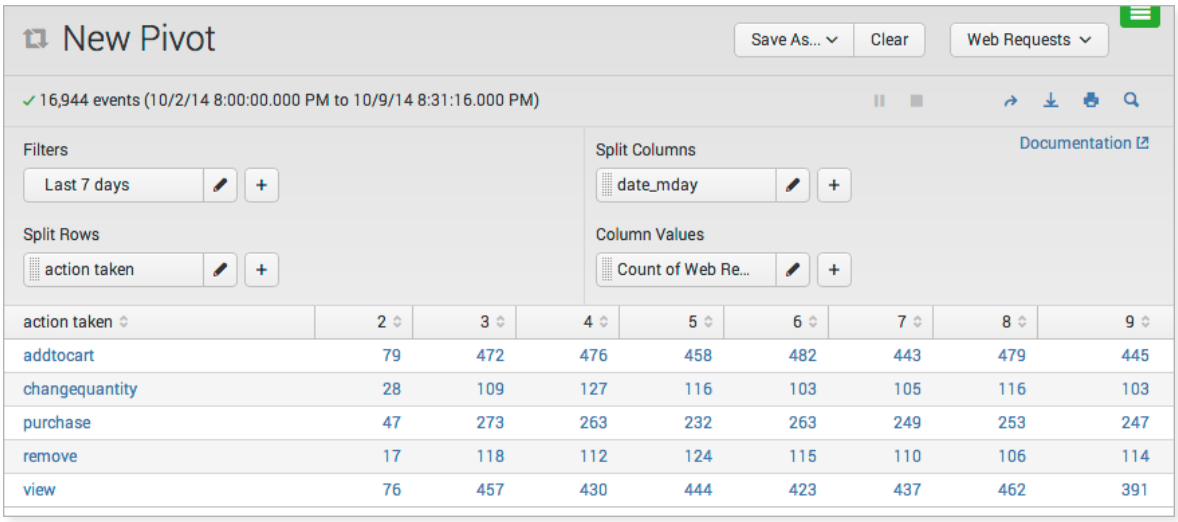
Under the Failed Requests object, add a child object called **removed** with the constraint of **`action=remove productId=*`**.

Results Example:



Task 4: Test your data model by creating a pivot from the Web Requests object. Filter on the **action taken** in the **last 7 days** and table the results by **date_mday**.

Results Example:



action taken	2	3	4	5	6	7	8	9
addtocart	79	472	476	458	482	443	479	445
changequantity	28	109	127	116	103	105	116	103
purchase	47	273	263	232	263	249	253	247
remove	17	118	112	124	115	110	106	114
view	76	457	430	444	423	437	462	391

Task 5: Add an attribute under the Web Requests object that uses an eval expression. The eval expression will table events chronologically by date and day.

Eval Expression = `strftime(_time,"%m-%d %A")`

Field Name = day

Display Name = day

Results Example:

Task 6: Verify the eval expression works as expected by using Pivot to create a dashboard. Filter on **actions** taken in the **last 7 days** and table the results by **day**. Create a dashboard panel called **Cart activity by day** and add it to the **Weekly Website Activity** dashboard.

Results Example:

Weekly Website Activity

Edit More Info

Cart activity by day <1m ago

action taken	05-07 Wednesday	05-08 Thursday	05-09 Friday	05-10 Saturday	05-11 Sunday	05-12 Monday	05-13 Tuesday	05-14 Wednesday
addtocart	89	114	117	111	119	145	133	12
changequantity	48	47	65	58	52	72	74	3
purchase	110	131	154	142	138	171	164	15
remove	35	64	59	80	76	77	59	3
view	181	229	247	303	266	254	288	25

Task 7: Add attributes from the **http_status_lookup** table to the Web Requests root object. Use the **code** field in the lookup table and map the attribute to **status** (this maps the `status` field in your indexed data to the `code` column in the lookup table.) For the lookup **Output** section in the **Field in Lookup** field, check the **description** checkbox. The display name is **status description.**

Task 8: Verify the lookup works properly by creating a Pivot report for the Web Request object. Split on the **status description** and **status** over the **last 7 days**.

Note: This is a double row split, not a column split.

Results Example:

status description	status	Count of Web Requests
Bad Request.	400	577
Forbidden.	403	182
HTTP Version Not Supported.	505	357
Internal Server Error.	500	533

Show a table of the results by day. Create a dashboard panel called **Web Requests Summary** and add it to the **Weekly Website Activity** dashboard.

Results Example:

Weekly Website Activity

Edit | More Info

Cart activity by day — 1m ago

action taken	05-07 Wednesday	05-08 Thursday	05-09 Friday	05-10 Saturday	05-11 Sunday	05-12 Monday	05-13 Tuesday	05-14 Wednesday
addtocart	79	114	117	111	119	145	133	15
changequantity	40	47	65	58	52	72	74	4
purchase	95	131	154	142	138	171	164	18
remove	30	64	59	80	76	77	59	4
view	163	229	247	303	266	254	288	36

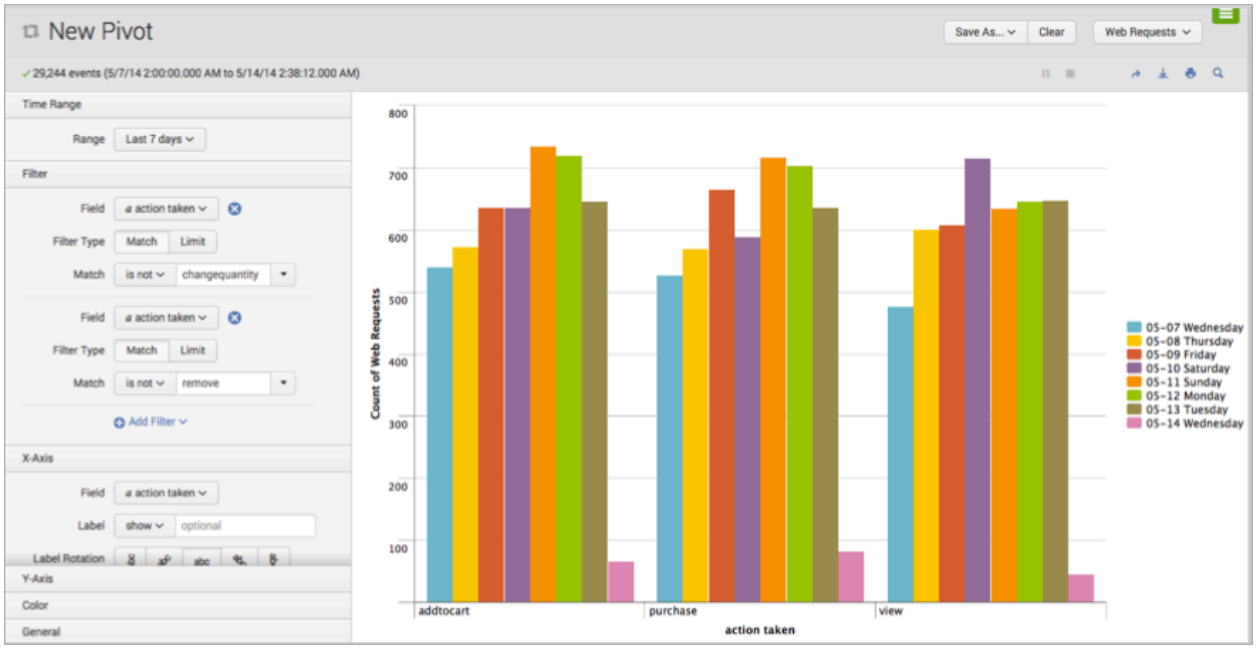
Web Requests Summary — 1m ago

status description	status	05-07 Wednesday	05-08 Thursday	05-09 Friday	05-10 Saturday	05-11 Sunday	05-12 Monday	05-13 Tuesday	05-14 Wednesday
Bad Request.	400	127	193	201	213	202	234	203	24
Forbidden.	403	50	69	60	66	73	71	75	6
HTTP Version Not Supported.	505	69	130	124	164	121	136	130	15
Internal Server Error.	500	113	175	206	188	209	238	224	19
Not Acceptable.	406	132	164	199	197	188	224	207	26
Not Found.	404	114	177	223	197	213	210	219	20
OK.	200	6371	9028	9779	10281	9952	10379	10860	1352
Request Timeout.	408	118	175	193	217	199	195	219	25
Service Unavailable.	503	190	245	265	278	276	298	309	47

Supplemental Exercise:

Task: From the pivot editor, add an attribute as a filter that displays all shopping cart activity except **changequantity** and **remove**. Create the report with columns.

Results Example:



Create a dashboard panel called **Add Purchase View** to the **Weekly Website Activity** dashboard.