Demystifying Machine Learning And Anomaly Detection:

Practical Applications in Splunk for Insider Threat Detection and Network Security Analytics

Toby Ryan

Emerson Incident Response Team (CIRT)

.conf2016

splunk>

Bio

- Current: Manager, Behavioral Analytics, Emerson Computer Incident Response Team (CIRT)
  - "Unofficial" Data Scientist
  - Serve as the design lead for our Splunk custom analytics platform
  - Manage the Insider Threat Program
  - Member - Carnegie Mellon CERT Open Source Insider Threat (OSIT) working group
  - Chair – OSIT Data Analytics Special Interest Group
  - Board of Advisors - Carnegie Mellon CERT Open Source Insider Threat (OSIT) working group

splunk> CERTIFIED ARCHITECT 6.3

- Prior to Emerson: Special Agent, US Naval Criminal Investigative Service (NCIS)
  - Insider Threat, Cyber, and Fraud Investigations (8 years)

- 1996-2007: The "Lost" Years

- BS in Spatial Information Science and Engineering – University of Maine (1996)

  (I was doing data science before it was cool!)

Goals of the Session:

- You will be able to describe the similarities and differences between internal/insider and external threats
- You will be able to map Machine Learning (ML) and Anomaly Detection (AD) algorithms to security use-cases
- You can start demystifying ML and AD by using practical security applications of ML and AD with Splunk Enterprise
- You will have the knowledge of where to start your own Security-Purposed ML and AD platform using Splunk Enterprise.
- You can start the conversation between technical experts and non-technical Insider Threat experts

Agenda

- Overview of threat types
- Data Science cycle for security
- Architecture of a Splunk-based Anomaly Detection platform
- Types of anomalies used in security use-cases
- Solving a security problem with Machine Learning
  - Deep dive for email analytics
  - Practical applications in ML
  - Anomaly Detection model improvement
  - Clustering for security
- Practical uses of ML and AD in various security and insider threat uses cases
- Advanced use-cases
- Wrap up and Questions

Why I Want To Talk To You….

- Insider Threat Programs are almost equally distributed between Human Resources, Legal, Security, and ***Information Security***
- That's roughly 75% that are **NOT** in a technical department
- If we are the 75%, how do we approach our Information Security departments to explain what we are looking for?
- If we are the 25%, how do we explain what we can do?

Highly Technical IT Security + **HR/Legal/Security** = **A Disconnect**

Internal vs. External Threats

- Insider Threat *categories*:
  - Malicious Insider
  - Non-Malicious Insider ~~(“Accidental Insider Threat”)~~
    – Negligent Insider
  - External actor *behaving* like an insider

- 3 *types* of Insider Threats:
  - Data Theft (Intellectual Property, PII, Financial, etc.)
  - Fraud
  - Sabotage

Alarming Statistics

- 62% Of employees think it is OK to move work documents to personal computers or mobile devices
- 51% Think it is OK to take corporate data because policies are not enforced; over half of employees surveyed who lost their job in the previous 12 months kept confidential data
- 56% Do not think it is a crime to use competitor's trade secrets

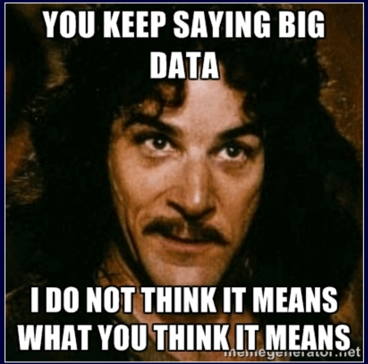
So…….. If you stood at the door on a Friday and stopped all resigning employees, you would have a 1 in 2 chance of catching somebody

Source: 2013 Symantec Global Survey – Insider Threat

What The Statistics Say - Generally

- Insider threats account for 25%-45% of cyber attacks
- Malicious Insiders steal data, commit fraud, or set the sabotage in action within ***the last 30 days of employment***
- Negligent Insiders are becoming the majority of insider threats
- 10%-20% Of employees click on malicious links in phishing emails
- Privileged users (Admin, DBA, IT Security, access to trade secrets, etc.) are companies' biggest concern

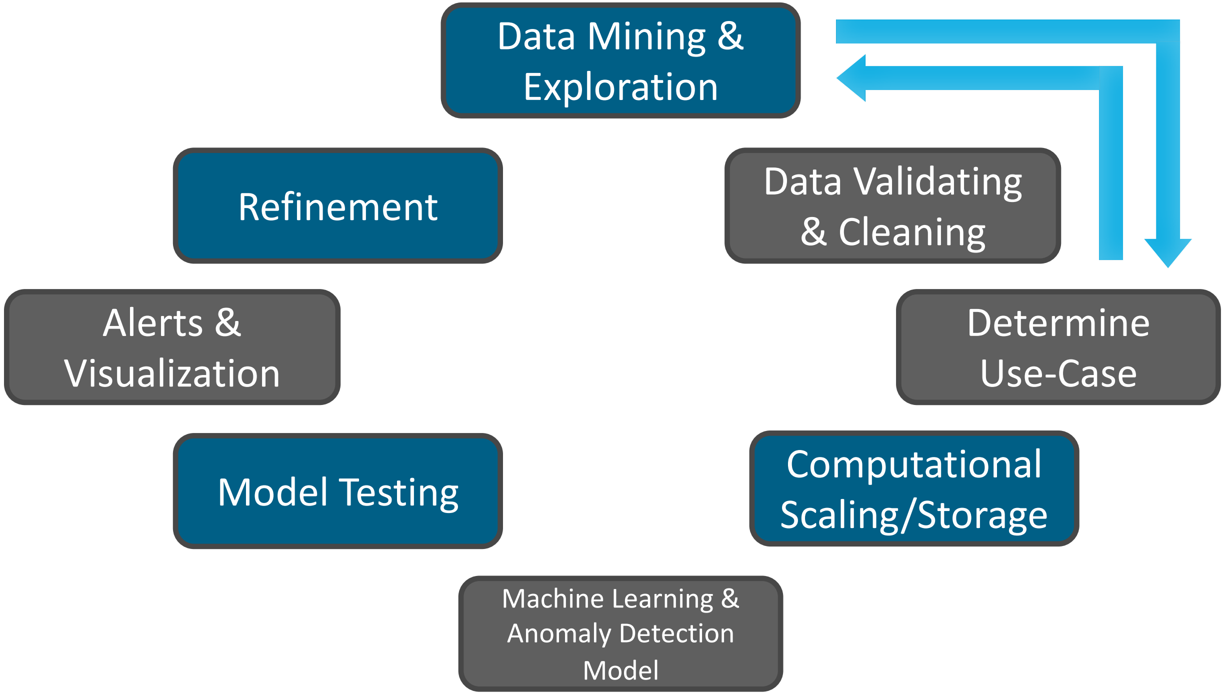
Let's Get Into The Data……



The Data Science Cycle For Security

- Determine Use-Case
- Data Mining & Exploration
- Data Validating & Cleaning
- Computational Scaling/Storage
- Machine Learning & Anomaly Detection Model
- Model Testing
- Alerts & Visualization
- Refinement

The Data Science Cycle For Security (V2)

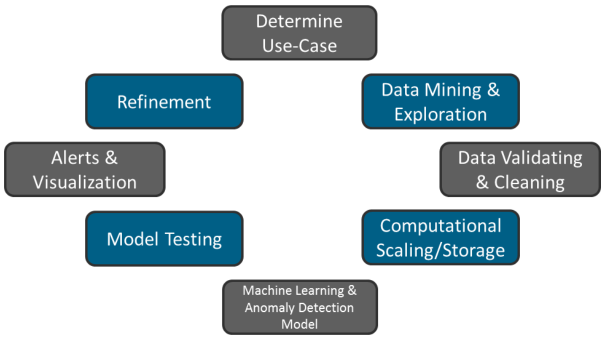


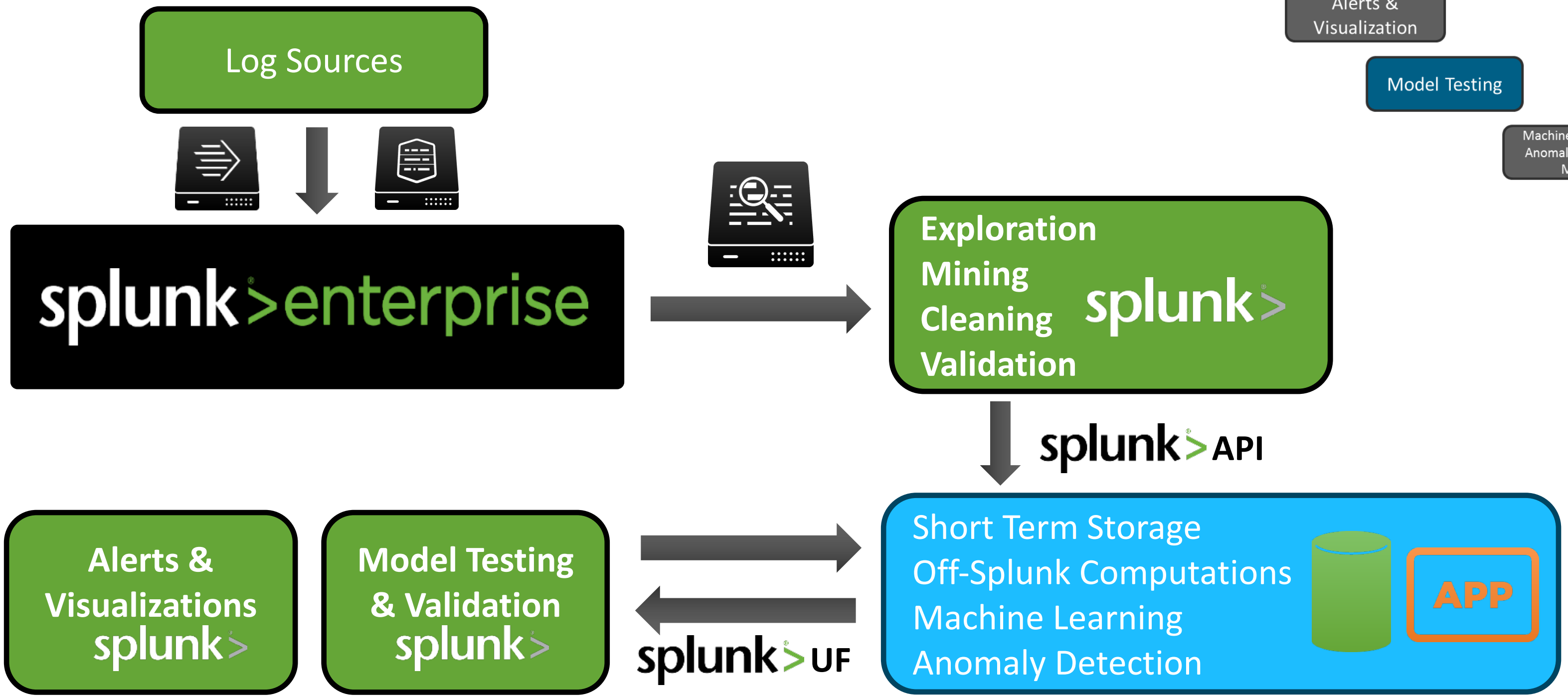
In Splunk Terms….

Determine Use-Case	Select Sourcetype(s)
Data Mining & Exploration	Verbose Mode Exploring
Data Validating & Cleaning	Compare Events/Table Data
Computational Scaling/Storage	API/Short-term Storage
Machine Learning & Anomaly Detection Model	What fields to measure: Numerical vs. Text
Model Testing	Do the results make sense?
Alerts & Visualization	Splunk Alerts/Dashboards
Refinement	Repeat Cycle

Architecture

Global enterprise environment with 100,000+ users/accounts





Off-Splunk Calculations at Scale

All are open source products

- Why move off-Splunk?
- Computationally Expensive
- Needs for Machine Learning and Anomaly Detection are different
- Splunk used for exploration and model development at testing scale
- Splunk Machine Learning Toolkit

Machine Learning Toolkit and Showcase

python™

GNU Octave

SciPy.org

ANACONDA

MySQL

elastic

scikit learn

R

Anomaly Detection & Machine Learning

- What is AD?
- Types of security anomalies:
  - spikes in activity
  - rare events
  - first-observed
  - outliers
  - state change
  - simple existence

What do these have in common?

time-based

The basic comparison parameter is *self-comparison over time*. Advanced parameters include peer-based comparison.

- What is ML?
  - Supervised ML
    - Classification/Regression
  - Unsupervised ML
    - Clustering
  - Semi-Supervised
    - Rule-based AD

For AD and security, ML can establish a baseline of normal (negative) values

Unsupervised Learning

- Unsupervised Machine Learning
  - You have unlabeled data and want to group the data by feature(s)
  - The algorithm makes its own structure out of the data
  - You do not know what outliers look like
  - Good for the data exploration phases of security anomaly detection
  - Examples used in security applications include:
    - Clustering: k-means, k-medians, Expectation Maximization
    - Association: less relevant because in highly structured searches we are less concerned with associations between fields for *security* anomaly detection

Supervised Learning

- Supervised Machine Learning
  - You have labeled data and the algorithm predicts the output
  - Classification vs. Regression
  - Example ML algorithms include:
    - Linear and Logistic Regression
    - Random Forest
    - Support Vector Machine
    - DBSCAN
- Semi-Supervised Machine Learning
  - You have “some” labeled data, but not all
  - Most security ML applications fall in this category
  - Label Propagation
  - Rule-based anomaly detection

For SECURITY-PURPOSED applications of ML, a combination of unsupervised, supervised, and Semi-Supervised learning algorithms is a best practice

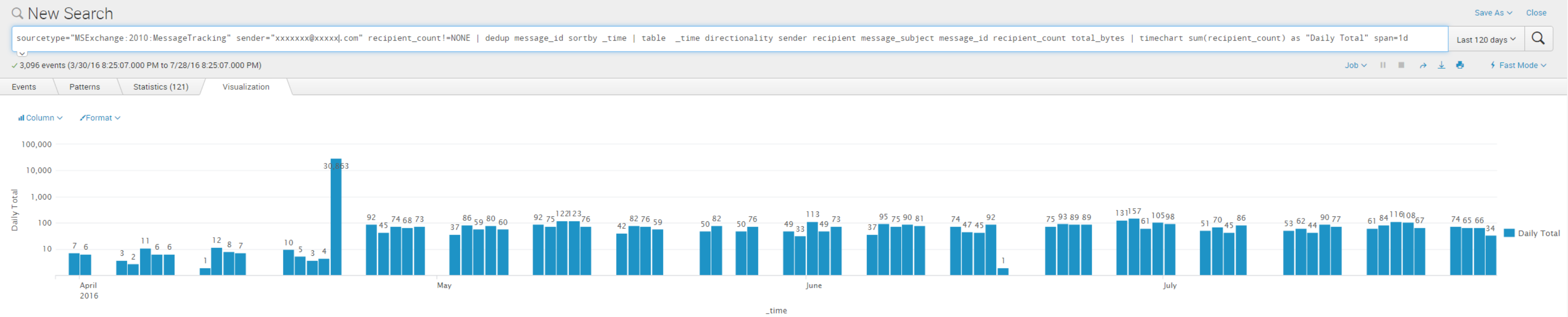
In realistic applications, security-purposed AD requires highly structured data and human training of the algorithm

Why Do We Use ML & AD?

- Growing ability of adversaries to avoid edge detection
- Signature-based tools only detect on a known signature- they must have an example or name of a bad “thing” to say what they are measuring is “bad”
- Behavioral-based detections target human nature
  - What is at the end on an “endpoint?”
  - What is the weakest link?
  - Targets computer and network behavior to determine what’s normal and what’s not
- Why do so many phishing emails get through?

> Human action can be modified to some extent, but human nature cannot be changed.
>
> (Abraham Lincoln)

Deep Dive:
Email Analytics for the Negligent Insider



Uh-oh…..

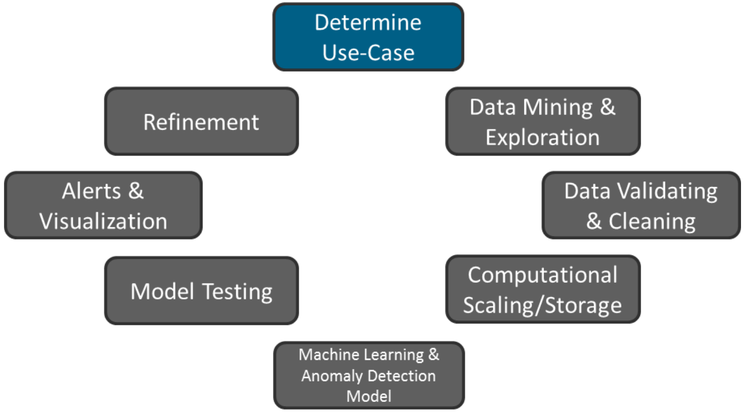
In this deep dive, we'll examine a compromise that would not be caught with traditional security stack tools but will be caught using basic ML & AD

Use-Case Deep Dive

Email Use-Case

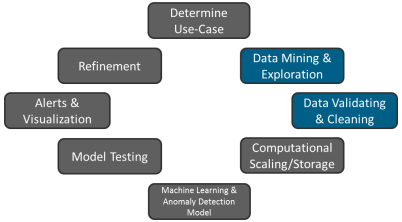
Use-Case Deep Dive

- Your company has been hit with a large number of phishing emails that were not detected by traditional signature-based tools
- Several employees have clicked on the phishing link and entered their credentials
- The adversary has taken over several accounts and sent thousands of additional emails, internal and external

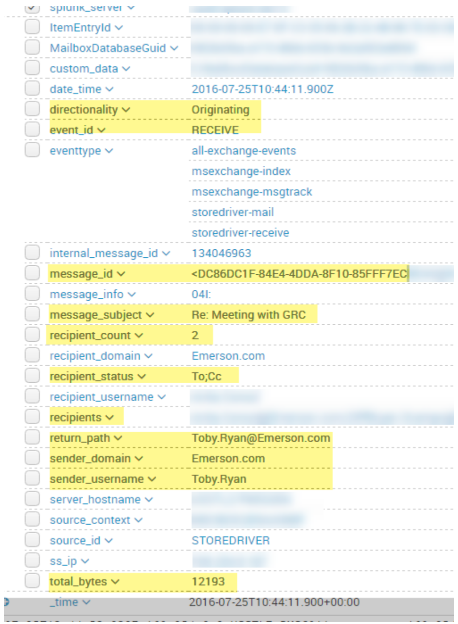


Data Mining & Exploration

Use-Case Deep Dive



- What looks interesting in this sourcetype?
- What could be used to detect an anomaly?
- What is important to note about the events?
- Send an email to yourself, then to a co-worker, then to several people, etc. as a validation test; trace the actions through Splunk

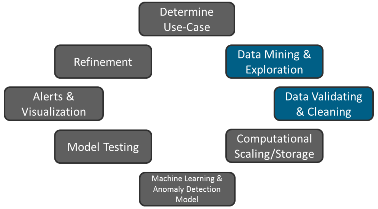


ML & AD for Security Best Practice:
Validate data by viewing your own actions on the network

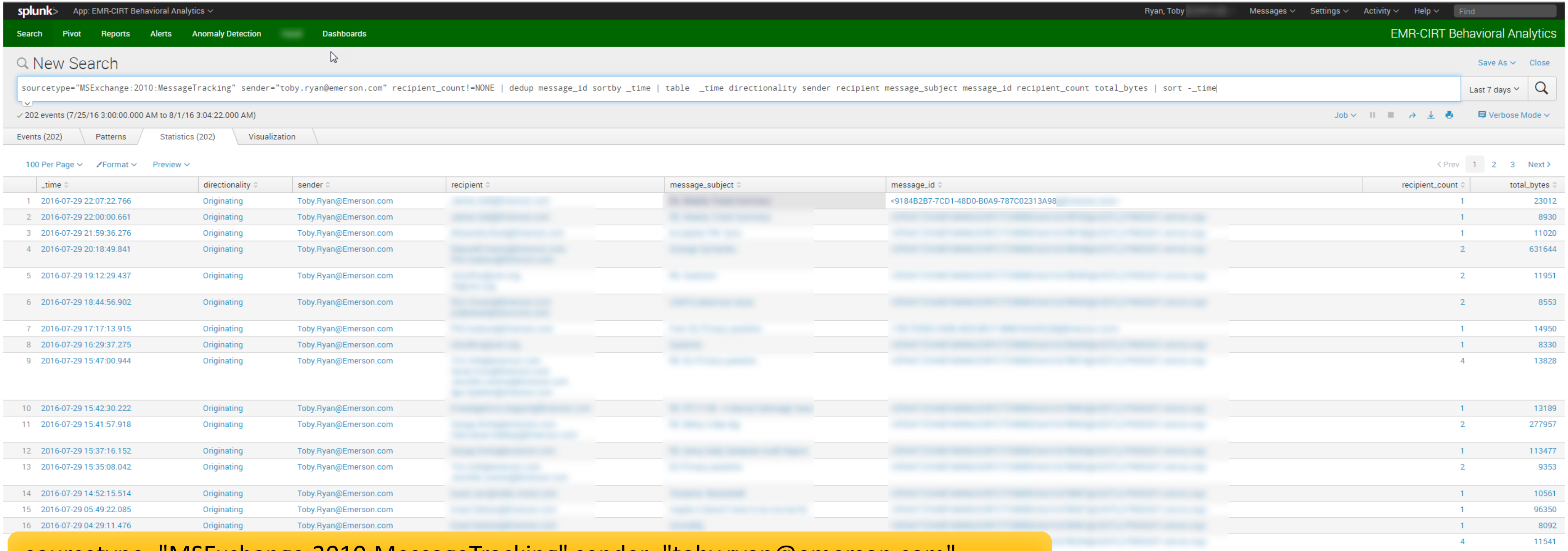
sourcetype="MSExchange:2010:MessageTracking"

Data Cleaning

Use-Case Deep Dive



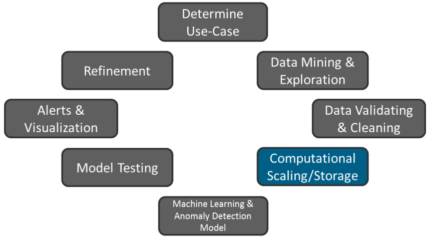
- What fields are best poised for measuring?
- What fields provide enough context for analysis?



sourcetype="MSExchange:2010:MessageTracking" sender="toby.ryan@emerson.com" recipient_count!=NONE | dedup message_id sortby _time | table _time directionality sender recipient message_subject message_id recipient_count total_bytes | sort -_time

Moving Data

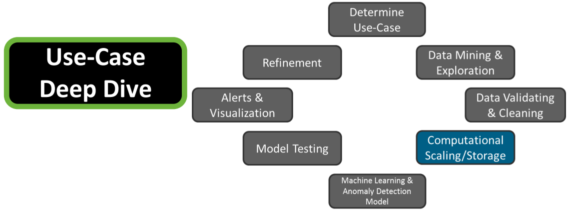
Use-Case Deep Dive

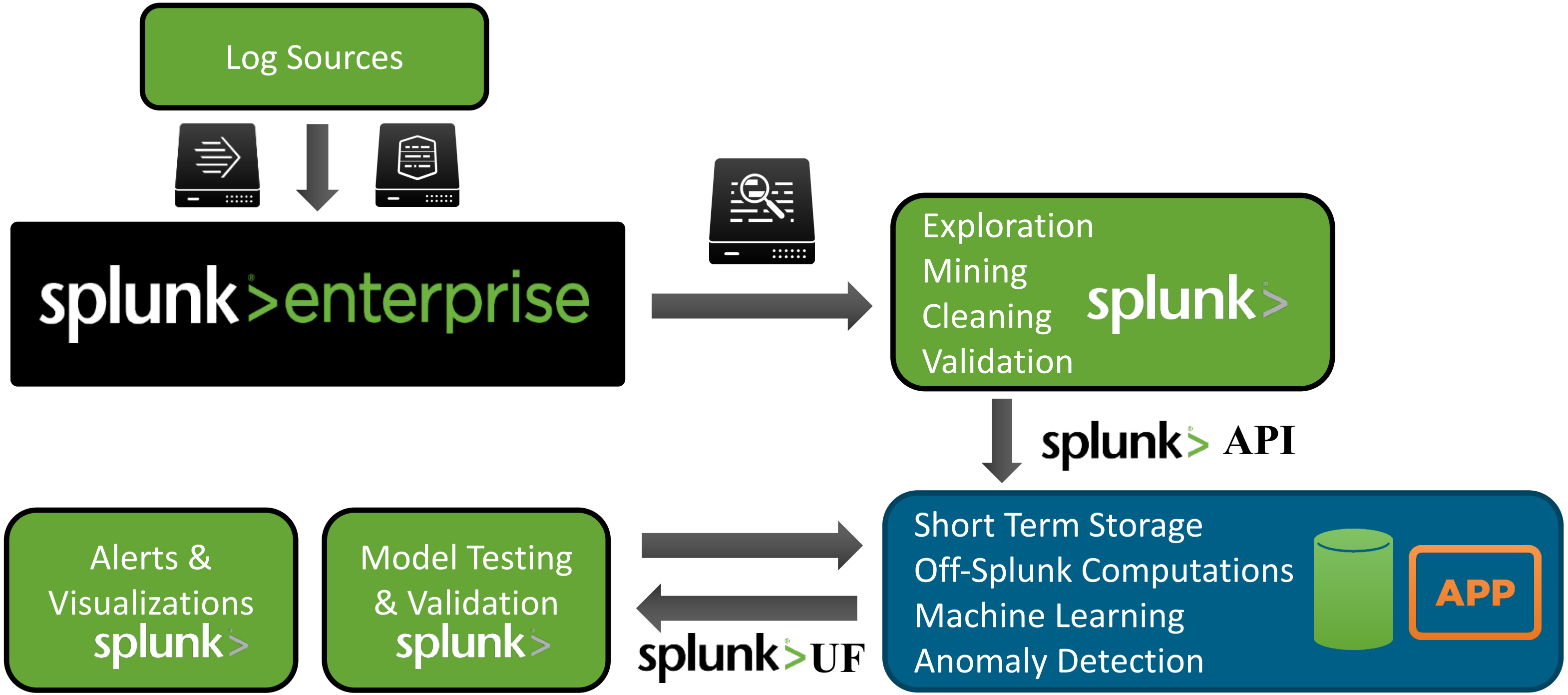



```
#! /usr/bin/env python
# Programmed by Grant Richard Steiner
# Last edited on 07/21/2016 - GS
# 07/19/2016 - Output to CSV
# 07/20/2016 - tranferred over to workstation (running centOS 7), set up cron job, queries, outputs to proper
#           directory
# 07/21/2106 - fixing permissions, mounted directory to linux machine, should output to correct directory
# import necessary libraries
import time
import subprocess
import datetime as dt
# to time the program, not necessary
START_TIME = time.time()
# for naming files
START_DATE = dt.datetime(2016, 01, 1)
#######################################################################################################
# This block gets data and pumps it to the necessary location
location = '/data/ws13/unique_email_averages_%s.csv' % str((dt.datetime.now() - START_DATE).days)
# Splunk search
search_string = '''search sourcetype=MSExchange:2010:MessageTracking sender=*@emerson.com recipient_count!=NONE ' \
          'earliest=-1d@d latest=-0d@d | dedup message_id sortby _time ' \
          '| fields _time sender message_subject message_id recipient_count ' \
          '| eval recipient_count = if(recipient_count=NULL, 0, recipient_count) | bucket _time span=1d ' \
          '| stats sum(recipient_count) as Daily_Total by sender, _time'''
# call to the cURL executable to run the search from the Splunk API
command_string = "/usr/bin/curl " \
          "-k -u cirt_insider:insiderTHREAT!!dash# " \
          "https://splunk.emrsn.org:8089/services/search/jobs/export " \
          "--data-urlencode search=%s -d output_mode=csv -o " \
          "%s" % (search_string, location)
# run the call to the Splunk API
subprocess.check_output(command_string, shell=True)
#######################################################################################################
print time.time() - start_time
```

- Utilize the Splunk API to call, move, transfer, or deposit data
- In this use-case, we are pulling over 120 days worth of data initially, and then pulling daily totals
- Data is moved to a short-term storage container (in this case, Elastic Search, but MySQL, or other open source SQL or NoSQL DBs work fine
- Why are we doing this?
  - Splunk's AWESOME integration capability
  - Computationally expensive within Splunk
  - Enterprise constraints

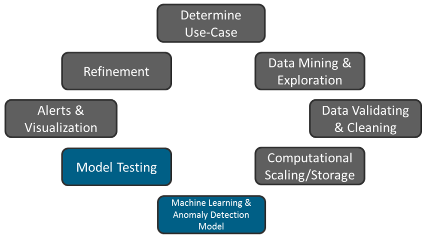
Where Are We In The Platform?





ML & AD Model

Use-Case Deep Dive



- What features do we choose?
- Supervised? Unsupervised? Classification?
- What statistical model do we choose?
- Start by clustering all data
  - Splunk “cluster” command for text and “kmeans” for numerical fields
  - What command is a clustering command in disguise?

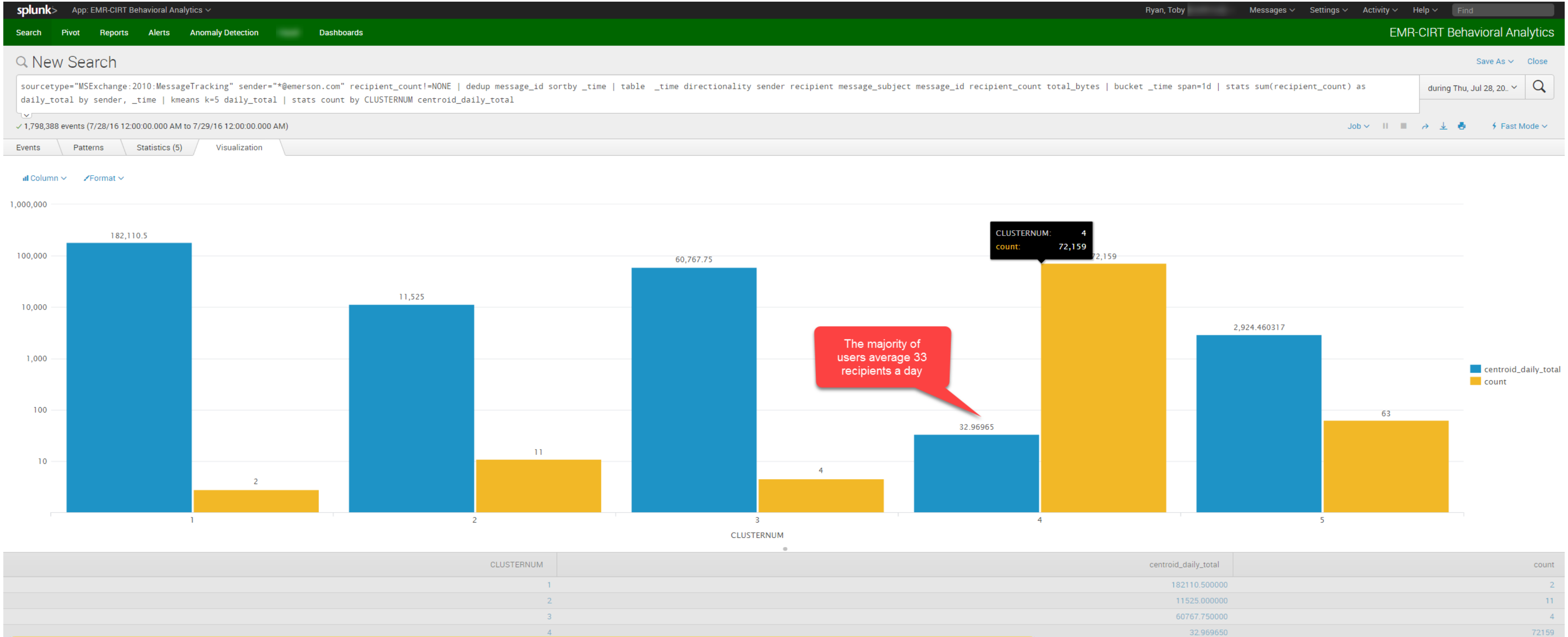
| stats count by {field being measured}

- Why do we cluster first?
- How many features do we choose?

ML & AD for Security Best Practice: From an incident response perspective, highly structured and single feature data is required to minimize time considering false positives

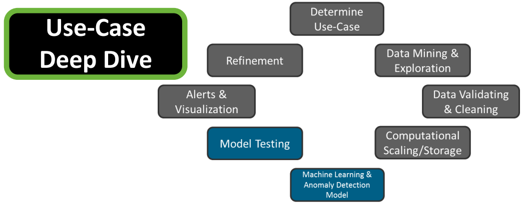
K-Means Clustering

Use-Case Deep Dive



sourcetype="MSExchange:2010:MessageTracking" sender="*@emerson.com" recipient_count!=NONE | dedup message_id sortby _time | table _time directionality sender recipient message_subject message_id recipient_count total_bytes | bucket _time span=1d | stats sum(recipient_count) as daily_total by sender, _time | kmeans k=5 daily_total | stats count by CLUSTERNUM centroid_daily_total

Training Data And The ML Process

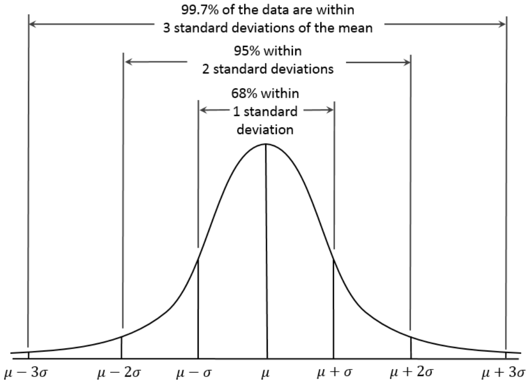


- Collect a set of training data (univariate/single feature/single field)
  - In our case, it is 60-120 days worth of daily email totals
  - Next, split the data by time into 3 groups: training set, cross-validation set, test set
- Determine if your dataset is Gaussian (Normal Distribution)

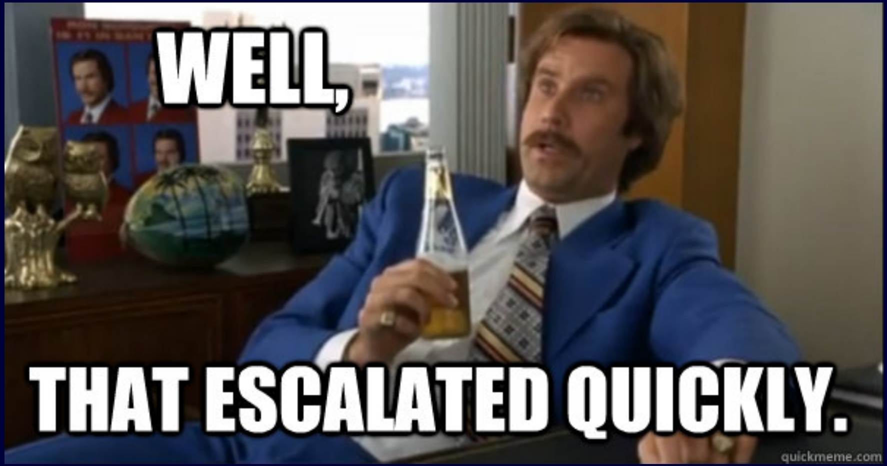
$$p(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sigma\sqrt{2\pi}}$$

ML & AD for Security Best Practices:

- Split historical data 60-20-20 into training, cross-validation, and test sets
- Don’t reuse data; do not use random splits



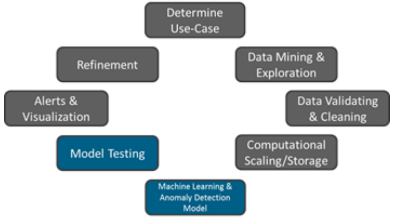

```
Total number of addresses analyzed: 80390
Total number of addresses discounted due to lack of viable data: 23521
Total number of addresses found to fit a normal distribution: 58790
Percentage of values discounted: 29.26%
The percentage of users with viable data that could be classified under a normal distribution is 73.13%
```



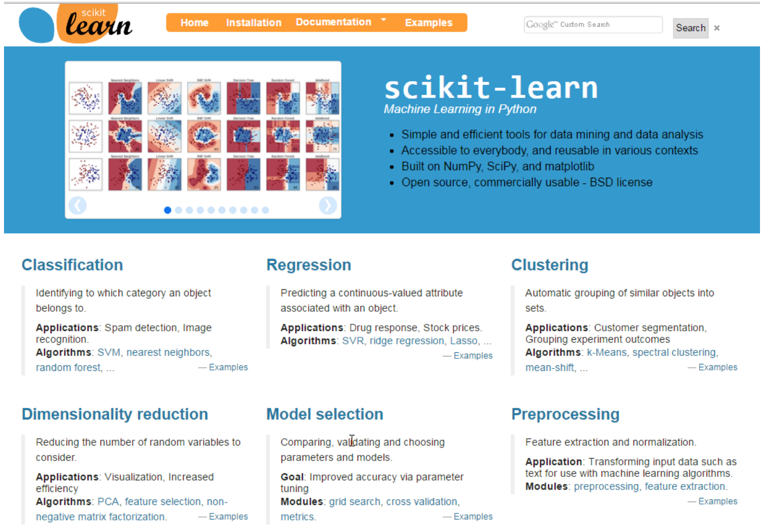
Don’t worry, Splunk and SciPy/Scikit do the math for you!

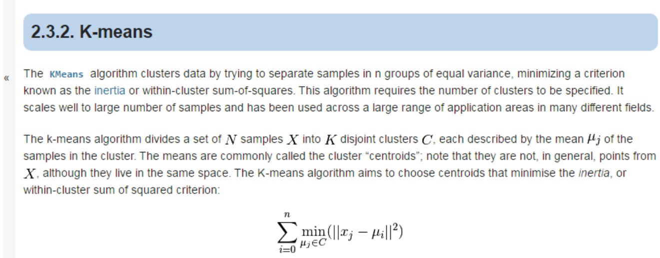
Algorithm Selection

Use-Case Deep Dive



- For normal distributions, Inter-Quartile Range (IQR) is a good place to start
- We can test back in Splunk for specific cluster users (using self-generated data)
- Other options available include:
  - Scikit-learn.org has the python modules
  - MATLAB, GNU Octave, and R all have extensive ML and AD packages
  - Python has easy Gaussian test algorithms (used in this example)
    - scipy.stats.mstats.normaltest
    - scipy.stats.shapiro
- Scikit-Learn has in-depth explanations of each algorithm and command descriptions such as “fit(x)” and “predict(x)”, etc.





2.3.2. K-means

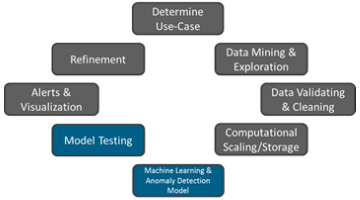
The `KMeans` algorithm clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares. This algorithm requires the number of clusters to be specified. It scales well to large number of samples and has been used across a large range of application areas in many different fields.

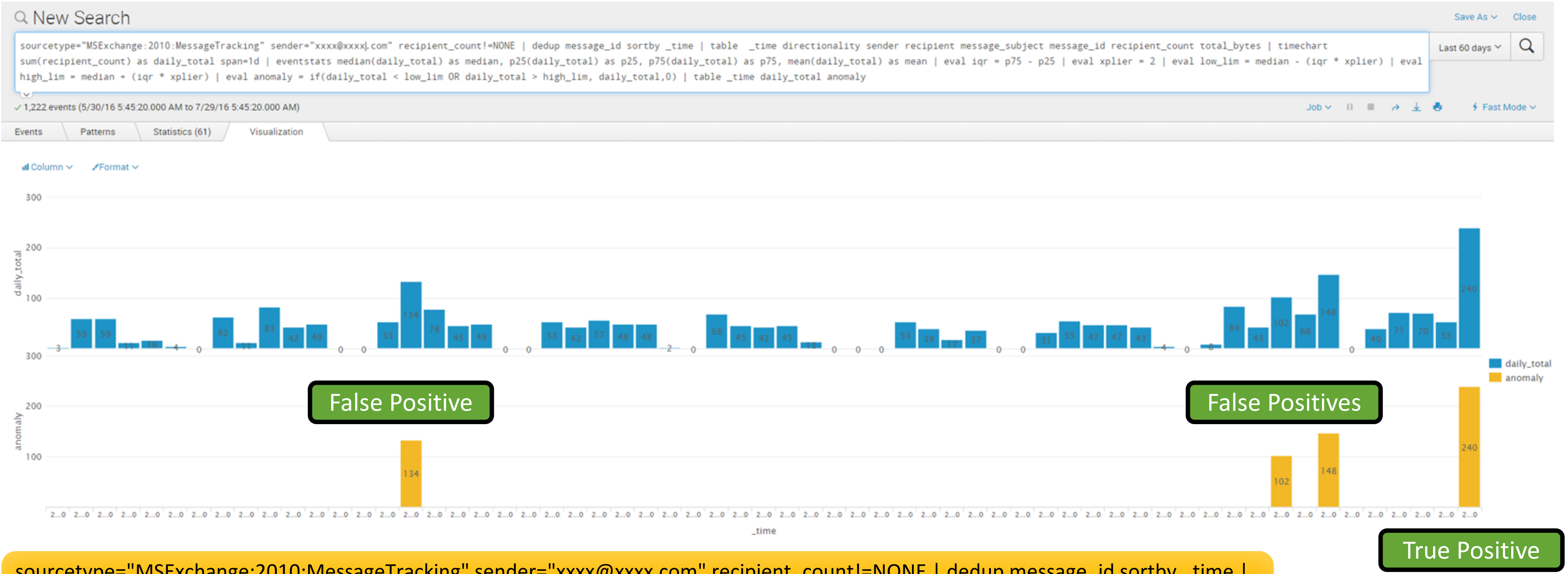
The k-means algorithm divides a set of N samples X into K disjoint clusters C, each described by the mean μ_j of the samples in the cluster. The means are commonly called the cluster “centroids”; note that they are not, in general, points from X, although they live in the same space. The K-means algorithm aims to choose centroids that minimise the *inertia*, or within-cluster sum of squared criterion:

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_j - \mu_i||^2)$$

Model Testing

Use-Case Deep Dive

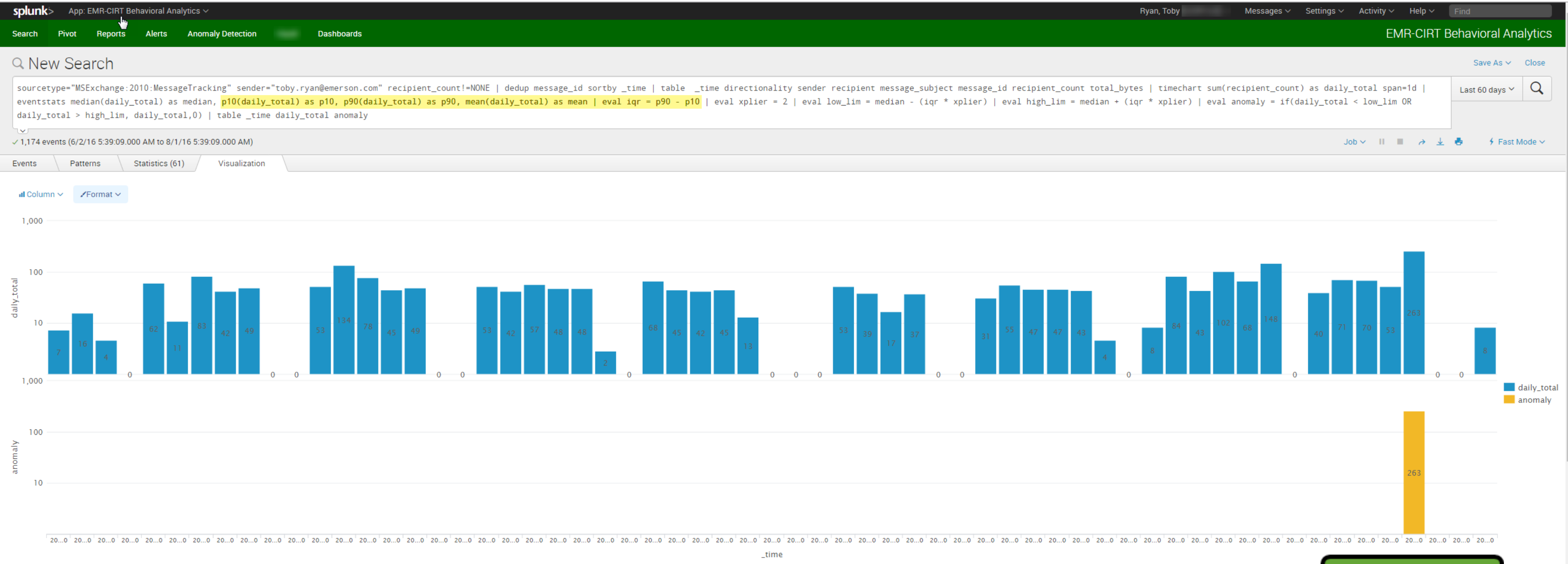




sourcetype="MSExchange:2010:MessageTracking" sender="xxxx@xxxx.com" recipient_count!=NONE | dedup message_id sortby _time | table _time directionality sender recipient message_subject message_id recipient_count total_bytes | timechart sum(recipient_count) as daily_total span=1d | eventstats median(daily_total) as median, p25(daily_total) as p25, p75(daily_total) as p75, mean(daily_total) as mean | eval iqr = p75 - p25 | eval xplier = 2 | eval low_lim = median - (iqr * xplier) | eval high_lim = median + (iqr * xplier) | eval anomaly = if(daily_total < low_lim OR daily_total > high_lim, daily_total,0) | table _time daily_total anomaly

Better?

Use-Case Deep Dive

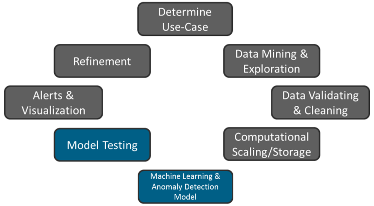


True Positive

sourcetype="MSExchange:2010:MessageTracking" sender="toby.ryan@emerson.com" recipient_count!=NONE | dedup message_id sortby _time | table _time directionality sender recipient message_subject message_id recipient_count total_bytes | timechart sum(recipient_count) as daily_total span=1d | eventstats median(daily_total) as median, p10(daily_total) as p10, p90(daily_total) as p90, mean(daily_total) as mean | eval iqr = p90 - p10 | eval xplier = 2 | eval low_lim = median - (iqr * xplier) | eval high_lim = median + (iqr * xplier) | eval anomaly = if(daily_total < low_lim OR daily_total > high_lim, daily_total,0) | table _time daily_total anomaly

Validating Models

Use-Case Deep Dive



- How can we validate models?

$$\text{Precision} = \frac{\text{\# of correct positive values}}{\text{\# of all positive results}}$$

$$\text{Recall} = \frac{\text{\# of correct positive values}}{\text{\# that should have been positive}}$$

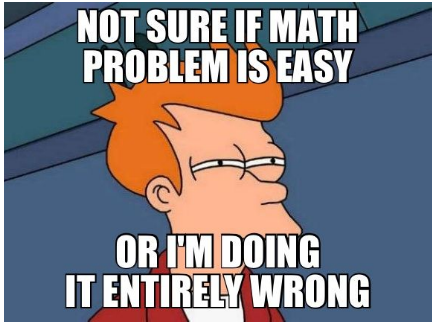
$$F_1 \text{ Score} = 2\left[\frac{\text{precision x recall}}{\text{precision + recall}}\right]$$

First model: $F_1 = 0.4$

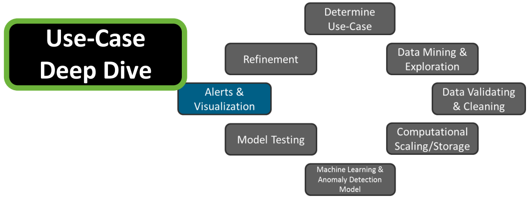
Second model: $F_1 = 1.0$

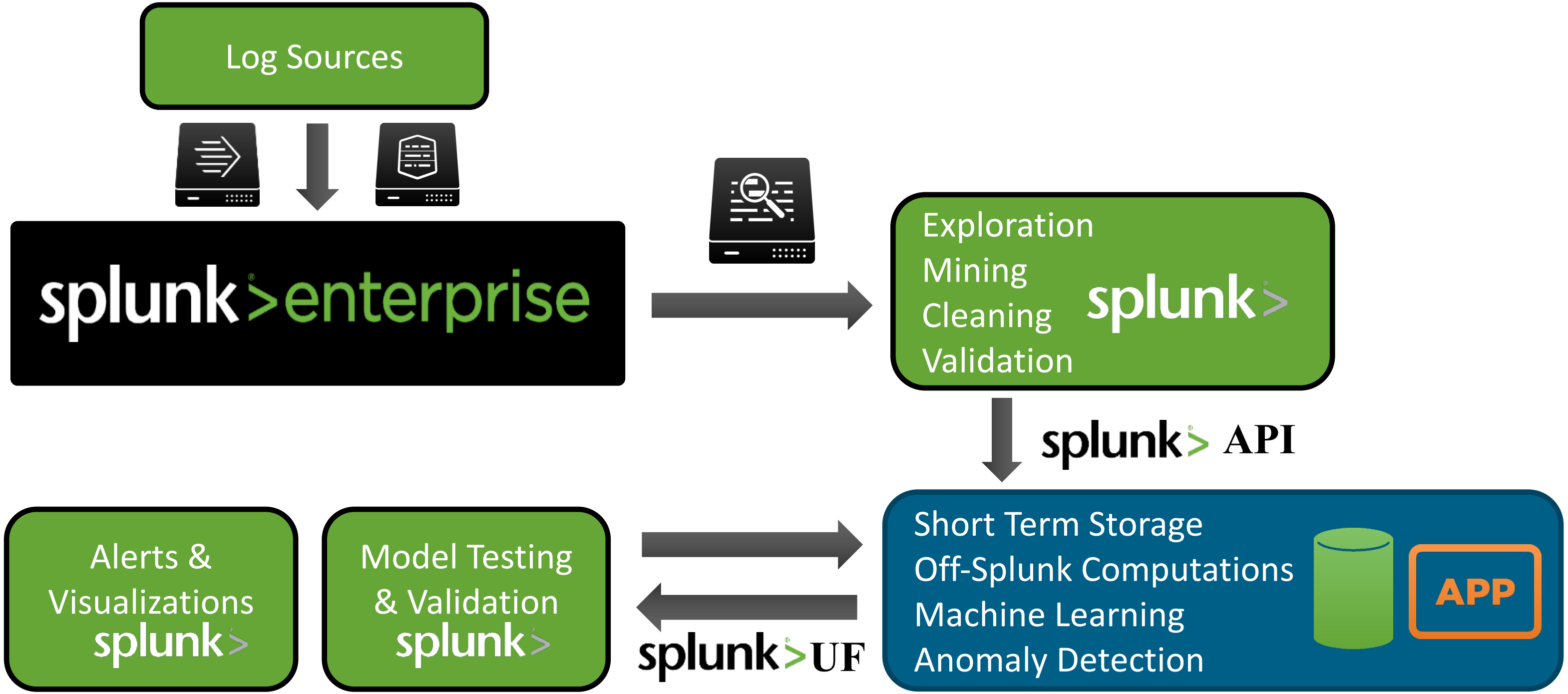
Beware of missing false negatives by tuning too much too quickly; tuning is an iterative process over time

F_1 Score is the harmonic mean, or average of rates, where F_1 is best at a value of 1, and worst at a value of 0.



Where Are We In The Platform?





One Last Note on Negligent Insider Email Analytics

Use-Case Deep Dive

- Consider not only a large number of recipients outside a user's normal behavior, but consider the *number of new recipients*
- What is the average number of new recipients an employee emails each day? One? Five?
- Establish a set of training data and record the unique recipients over 60 days
- Create an anomaly detection that fires when the number of new recipients exceeds the baseline variance
- Add to the "# of recipients per day" data for higher fidelity alerts
- Example:
  - Baseline number of daily recipients = 30
  - Today's amount = 75, but falls on the fringe of being an outlier
  - Number of new recipients = 1-5; false positive
  - Number of new recipients = 50; true positive

Move The Data To Short-term Storage For Measurement

Use-Case Deep Dive

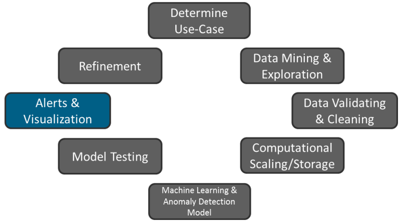
```
# !/usr/bin/python
# Programmed by Grant Richard Steiner
# Last edited on 07/28/2016 - GS
# 07/28/2016 - Generated first round of data for this query
# import necessary libraries
import time
import subprocess
import datetime as dt
import operator
import csv
# to time the program, not necessary
start_time = time.time()
# for naming files
START_DATE = dt.datetime(2016, 01, 1)
##########################################################################################################
# FUNCTIONS
"""
for de-duping a list
O(1) insertion, deletion and member-check per operation
"""
def dedup_list(seq):
    seen = set()
    seen_add = seen.add
    return [x for x in seq if not (x in seen or seen_add(x))]
##########################################################################################################
##########################################################################################################
# THIS BLOCK QUERIES THE SPLUNK API AND RETURNS THE RESULTS AS CSV (NOT FORMATTED FOR ES)
IN_FILE_PATH = 'c:\daily_recip_count_%s.csv' % str((dt.datetime.now() - START_DATE).days)
OUT_FILE_PATH = 'c:\'email_contact_history_%s_00.csv' % str((dt.datetime.now() - START_DATE).days)
# Splunk search
search_string = '"search sourcetype="MSExchange:2010:MessageTracking" sender="*emerson.com" recipient="*@emerson.com"' \
        ' recipient_count!=NONE earliest=-1d@d latest=-0d@d | dedup message_id sortby _time ' \
        '| fields _time sender recipient_count recipient recipients ' \
        '| table _time sender recipients"'
# call to the cURL executable to run the search from the Splunk API
command_string = "C:\\Users\bin\\curl.exe " \
        "-k -u cirt_insider:insiderTHREAT!!dash# " \

        "--data-urlencode search=%s -d output_mode=csv -o " \
        "%s" % (search_string, IN_FILE_PATH)
# run the call to the Splunk API
subprocess.call(command_string)
##########################################################################################################
##########################################################################################################
# THIS BLOCK RE-FORMATS THE CSV FROM THE SPLUNK CALL SO IT CAN BE PUMPED TO ELASTIC SEARCH
with open(IN_FILE_PATH, 'r') as raw_csv:
    # skips the field names in the raw_csv
    for_reading = csv.reader(raw_csv)
    next(for_reading)

    # sorts CSV by 'sender' field
    sorted_csv = sorted(for_reading, key=operator.itemgetter(2))
```

```
  # temp is an empty list that comparisons are made against
    # format is ['timestamp', 'recipient_string', 'sender']
    temp = ['', '', '']

    # data is a dictionary that will hold lists of recipients as values to dictionary keys, which are email addresses
    data = {}

    for row in sorted_csv:
        # if the third item in temp is not the third item in the row, we make temp the current row and create a new
        # key in the data dictionary and add the list of recipients
        if row[2] != temp[2]:
            temp = row
            data[row[2]] = row[1].split(';')
        else:
            # if they are the same, we simply add the list of recipients on the current row to the list of recipients
            # in the python dictionary
            data[row[2]] += row[1].split(';')

# writes the data in teh correct format to a csv
with open(OUT_FILE_PATH, 'wb') as for_ES:
    spamwriter = csv.writer(for_ES)
    for key in data:
        # does a quick dedup on each list before writing to the rows
        data[key] = dedup_list(data[key])
        """
        writes each item (recipient) in the list and the key (sender) in the following format:

        ...
        'sender', 'recipient1'
        'sender', 'recipient2'
        etc.
        ...

        """
        for item in data[key]:
            spamwriter.writerow([key, item])

print time.time() - start_time
```

Now in a position to compare # of new recipients

Alerts & Visualizations

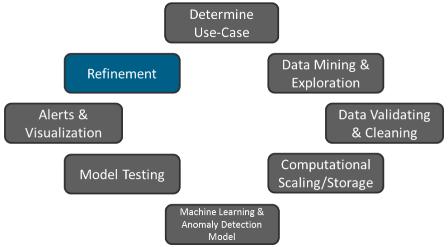
Use-Case Deep Dive



- The output of the off-Splunk calculations can be picked up by the Splunk UF or written to a flat file
- Allows the user to capitalize on the Splunk interface
- Advantages/Disadvantages of Indexing and Sourcetyping:
  - Treat like any other data source for calculations
  - Technically “re-indexing” data, however anomaly data sets will be small

Refinement

Use-Case Deep Dive



- Treat different clusters with different models
- Continually validate data and results
- Understand why false positives come up
- Add length to training data time if possible
- If a cluster is not Gaussian, try other models, or try to fit the data to a Normal Distribution
- Compare simple rule-based models such as 3 x mean = anomaly

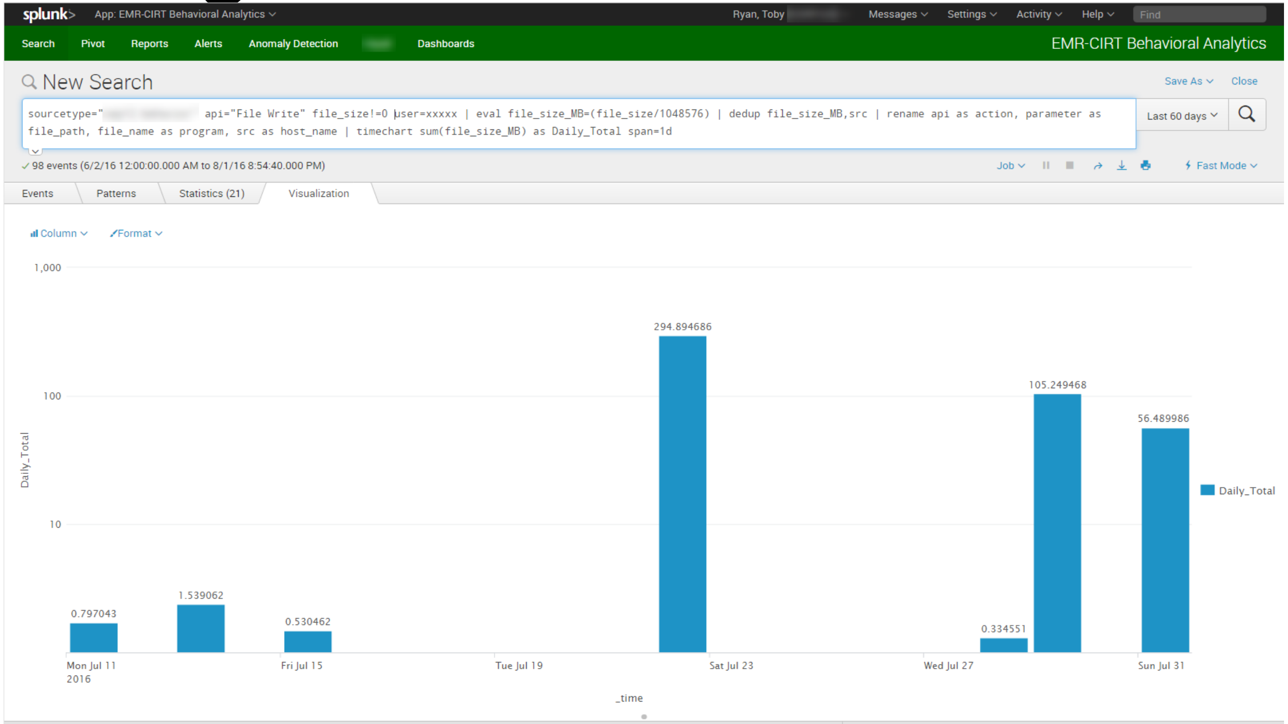
Additional Use-Cases & Use-Case Starter Searches

Use-case: Cross-correlation Of Departing Employees Through Rule-based Searches

- Cross-correlate USB activity with departing employees for insider threat detection
- Background: Historical data from insider threat incidents indicate large transfers of data prior to departure
- Data Source:
  - Endpoint Agent Logs (McAfee, Symantec, Kaspersky, etc.)
  - Message Tracking Logs

Use-case: Cross-correlation Of Departing Employees Through Rule-based AD Searches

- Most USB activity will not be a normal distribution
- Utilize K-Means to determine users who back-up their machines via USB
- Must utilize a rule-based approach
- Set a daily threshold such as: daily_total > 20 MB to indicate large data transfers



sourcetype="endpoint agent" api="File Write" file_size!=0 user=xxxxx | eval file_size_MB=(file_size/1048576) | dedup file_size_MB,src | rename api as action, parameter as file_path, file_name as program, src as host_name | timechart sum(file_size_MB) as Daily_Total span=1d

Use-case: Cross-correlation Of Departing Employees Through Rule-based AD Searches

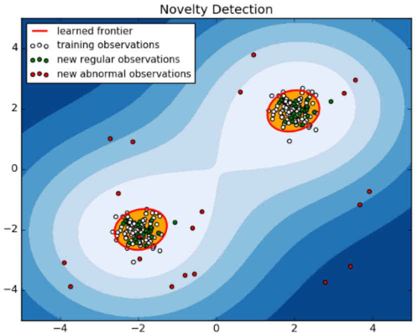
- Create a search that includes data from your HR organization OR… learn who is departing through a rule based search of email subjects
  - "*termination*"
  - "*resignation*"
  - Auto generated Oracle or Peoplesoft emails
- Combine the results with the spikes in USB activity search to create a two-feature classification learning algorithm

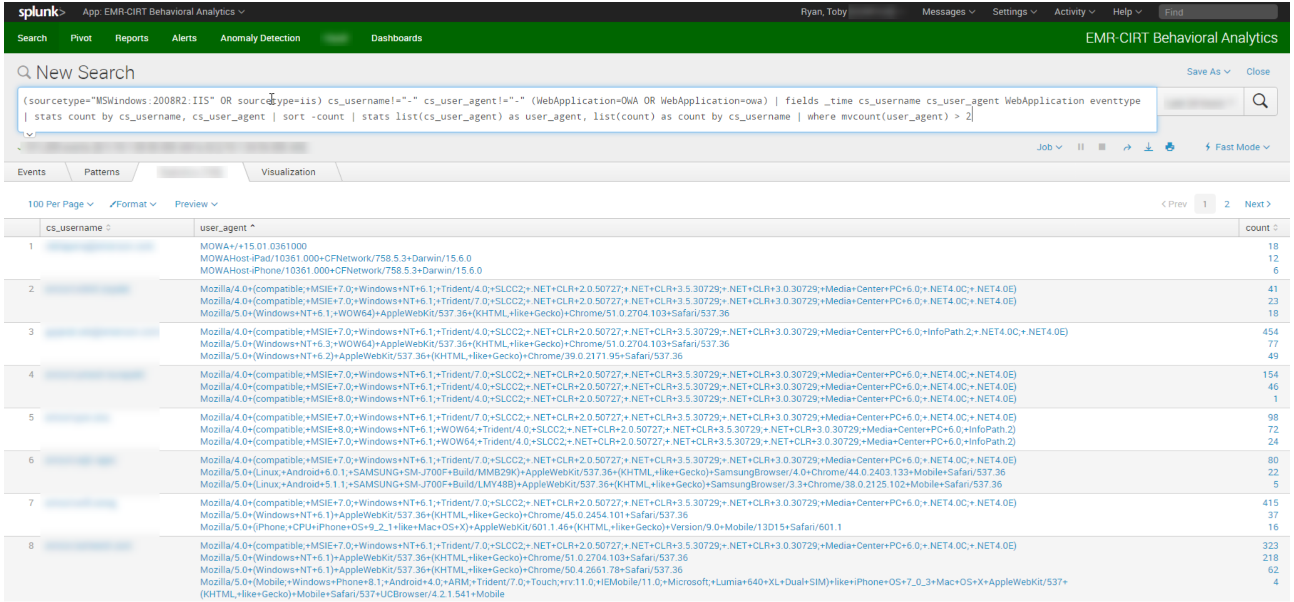
ML & AD for Security Best Practice: Clean and reduce size of the dataset to include only those items of value

```
sourcetype="MSExchange:2010:MessageTracking"  recipient_count!=NONE  message_subject!="*out of office*"
message_subject!="Automatic reply*" message_subject!="Customer Satisfaction*" message_subject!="READ:*"
message_subject!="*determination*" (message_subject="*resignation*" OR message_subject="*termination*") | dedup
message_id sortby _time | table  _time directionality sender recipient message_subject message_id
```

Use-Case: Email Client Type State Change

- Detect rare or anomalous values in client types used by Outlook Web Access (OWA) or Outlook Anywhere users as a sign of compromise
- Rarity or "novelty-based" anomaly detection over time
- First-observed detection



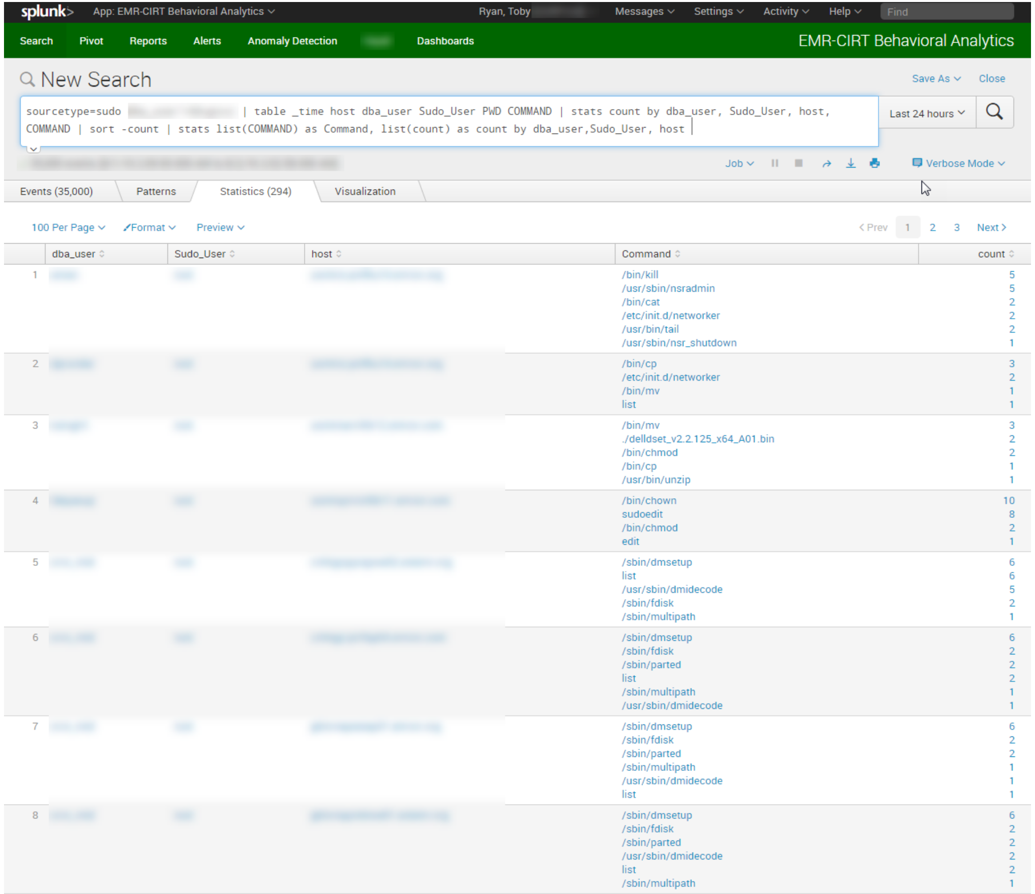


(sourcetype="MSWindows:2008R2:IIS" OR sourcetype=iis) cs_username!="-" cs_user_agent!="-" (WebApplication=OWA OR WebApplication=owa) | fields _time cs_username cs_user_agent WebApplication eventtype | stats count by cs_username, cs_user_agent | sort -count | stats list(cs_user_agent) as user_agent, list(count) as count by cs_username | where mvcount(user_agent) > 2

Use-case: Sudo Logs And Sabotage

- Look for anomalous patterns in sudo to root privileges
  - Time-based logins
  - Unauthorized scripts
  - Data Theft
  - Unauthorized server access

- Use a combination of supervised rule-based detection for script execution and time-based anomaly detection for authentication data



sourcetype=sudo dba_user | table _time host dba_user Sudo_User PWD COMMAND | stats count by dba_user, Sudo_User, COMMAND | sort -count | stats list(COMMAND) as Command, list(count) as count by dba_user,Sudo_User

Insider Threat Use-Case Starter Searches

- Determine anomalous values surrounding privileged windows admins who utilize RDP
  - Determine model for baseline – will probably not be a normal distribution
  - Set detections for values outside of baseline
  - Features include destination servers, time-of-day, and multiple novelty events

```
sourcetype="WinEventLog:Security" (EventCode=4624 OR EventCode=4625)  Logon_Type=10 user="*admin*" | table _time user Account_Name Workstation_Name ComputerName src_ip
```

- Use classification and supervised learning to detect sensitive file types in USB traffic
  - CAD files, financial documents, engineering, business intelligence, pricing, etc.
  - Set positive values to file extensions of interest such as .DWG

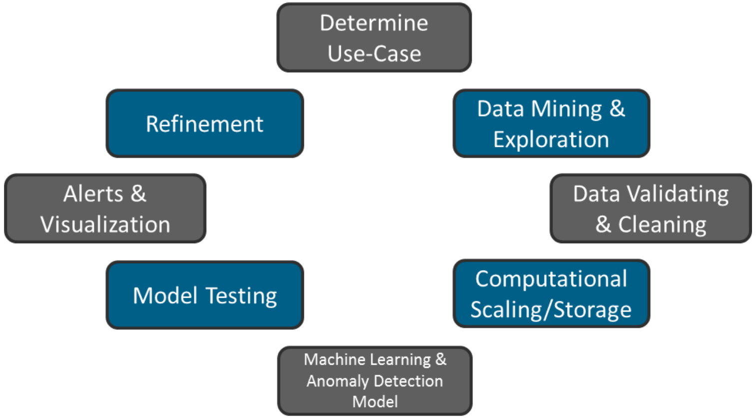
```
sourcetype=“endpoint agent" "File Write" file_size!=0 user=xxxxx  “*.DWG” | eval file_size_MB=(file_size/1048576) | dedup file_size_MB,src | stats count by src {or user}
```

Internal And External Threat Use-case Ideas

- FTP Servers: clustering IP addresses, frequency spike, rule-based detections using company-specific criteria (IIS/FW/LB Logs)
- Phishing and fraud email detection: domain mismatch using email metadata (message_id) to compare sending domain, display name, and return path (Message Tracking Logs)
- Large Number of file downloads/views/prints from application housing sensitive documents (App-specific and/or IIS logs if web-based)
- Anomalous port activity (Ports 53, 25, 21, 22, 443, 123, etc.)
- Authentication anomalies: login to a rare or first-observed device, off-hour login, pattern of single failed logins from several machines or Sharepoint locations (the “probing” user)
- Detecting shared credentials – especially among sensitive users (DBAs, Admins, etc.)

Wrapping Up - What Have We Covered?

- The Data Science Cycle for Security
- Deep Dive into ML & AD for security
- Demystified the math behind ML & AD and provided simple solutions such as classification algorithms
- Various Use-Cases for security ML & AD



The process of cleaning and carefully selecting data is more important than choosing the right algorithm

Questions?


Celebrate. Challenge. Consider It Solved.
Thank You
Emerson Electric Co.
Ben Davis
Security Engineer, Emerson CIRT
Grant Steiner
Engineering Co-Op, Emerson CIRT
EMERSON

THANK YOU