

SEC1305: Detecting and Mitigating Insider Threats Using MLTK and Enterprise Security

.conf19

splunk>



Karthik Subramanian

Principal Senior Cybersecurity Architect | SAIC



Tyler Williams

Principal Cybersecurity Data Analyst | SAIC

Forward-Looking Statements



During the course of this presentation, we may make forward-looking statements regarding future events or plans of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results may differ materially. The forward-looking statements made in the this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, it may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements made herein.

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

Splunk, Splunk>, Turn Data Into Doing, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2019 Splunk Inc. All rights reserved.



Insider Threat Detection with MLTK

“An insider threat is an individual who uses authorized access to an organization's assets to use their access, either maliciously or unintentionally, in a way that could negatively affect the organization.”

Carnegie Mellon CERT Insider Threat Center

Types of Insider Threat

Data Theft



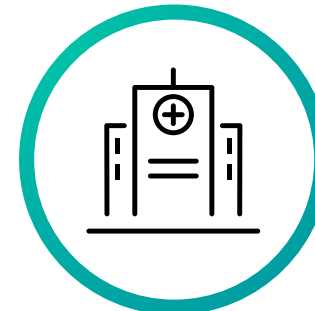
Sabotage



Fraud



**Workplace
Violence**



Espionage





What can we do about it?

Example Insider Threat Heuristics

What can we monitor to identify threats?

Local Logs

- In many Insider Threat cases, the activity begins with a user logging in at an abnormal time.

Network Logs

- Traffic Flow through your web proxy, for example can tell us a lot about potential Data Exfiltration.



But First....

Some things to consider about execution

Designing Use Cases

- Introducing use cases can be complicated
- Primary Behavior + Enriching Details
- Adaptable solutions



Use cases should be well-tested, adaptable, and policies should be understood by all stakeholders.

Notifying the SOC Analysts

- Dashboards with Drill-downs
- Alerts via email
- Notable Events



Measurements of behavior are established over a 30-day period which makes for some interesting alerting.



Using Your Data to Measure Insider Threat Behaviors

Some SPL code to identify a few important examples of user behavior heuristics

Example Use Case: Variance in Web Uploads

Based on true events

Scenario: An employee of a bank became concerned after a recent wave of layoffs. The user began searching for a new job. Before resigning, they capture a large amount of data from the bank's CRM and copy it into Word documents before emailing them during non-working hours and sending them out through a personal Gmail account.

- So in this case, is 20MB of Data outbound through the Web Proxy a threat to the organization? Yes, of course.
- How could you identify events like this in your environment?
- Signature-based policies won't work very well here. Try implementing an alert every time someone uploads more than 20MB of data in your network over the course of an extended period of time (But don't, actually, unless you hate your analysts).

Variance in Login Time

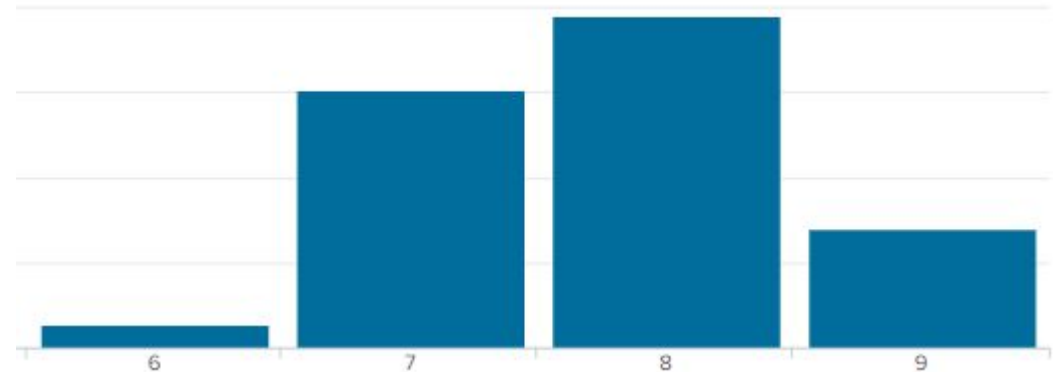
Our user logged in on a Saturday evening

How would we identify this type of behavior?

- Day of the week
- Time of Day
- By User

How would we measure variation from the baseline?

- Earliest and Latest Logins
- Statistical Measurements
- Account for employees on any shift



Login Activity Baselines

```
1 index="authentication" action=success earliest=-90d latest=-30d
  | eval val1 = sin(2 * pi() * (_time%86400)/86400),
  val2 = cos(2 * pi() * (_time%86400)/86400)
2 | stats avg(val1) as avg_sin,
  avg(val2) as avg_cos by user, date_wday, date_mday, date_month
3 | eval avg_time = atan2(avg_sin, avg_cos)
4 | stats avg(avg_time) as avg_time, stdev(avg_time) as stdev_avg_time by user, date_wday
  | outputlookup user_authentication_baseline.csv
```

Login Activity Baselines Search

Let's explain what's going on in that search

1. Get your time value to the number of seconds in a day, then make it into a circle because the difference between 11:59PM and 12:01 AM is 2 minutes, not 23 hours and 58 minutes.
2. Calculate the average modified time values by each day, including week day.
3. Obtain an arctangent value and take baseline descriptive statistics by user and day.
4. Output baselines for future use.

Login Activity Variation

```

1 | index="authentication" action=success earliest=-30d latest=now
   | stats min(_time) as early_login, max(_time) as late_login by user, date_wday, date_mday, date_month
   | eval val1_early = sin(2 * pi() * (early_login%86400)/86400),
   |   val2_early = cos(2 * pi() * (early_login%86400)/86400)
   | eval val1_late = sin(2 * pi() * (late_login%86400)/86400),
   |   val2_late = cos(2 * pi() * (late_login%86400)/86400)
2 | eval earliest_login = atan2(val1_early, val2_early),
   |   latest_login = atan2(val1_late, val2_late)
   | lookup user_authentication_baseline.csv user, date_wday
3 | OUTPUTNEW avg_time, stdev_avg_time
   | eventstats avg(stdev_avg_time) as fillnull_stdev by user
   | eval stdev_avg_time =
4 | if(stdev_avg_time=0 OR ISNULL(stdev_avg_time),fillnull_stdev, stdev_avg_time)
   | eval New_Day = if(ISNULL(avg_time),1,0)
   | eventstats avg(New_Day) as avg_new_day,
5 | stdev(New_Day) as stdev_new_day
   | eval Z_New_Day_Authentication = (New_Day-avg_new_day)/stdev_new_day
   | eval Z_Authentication_Risk_Score = max(abs((earliest_login-avg_time)/stdev_avg_time),
6 |   abs((latest_login-avg_time)/stdev_avg_time))
   | eval Z_Authentication_Risk_Score = COALESCE(Z_Authentication_Risk_Score, Z_New_Day_Authentication)
7 | stats max(Z_Authentication_Risk_Score) by user
   | outputlookup User_Behavior_Authentication_Risk_Score.csv

```

Login Activity Variance Search

Let's explain what's going on in that search

1. Calculate the earliest and latest login times for each user and day while retaining day of week values.
2. Calculate the modified time values with arctangent.
3. Take data from baselines and append to the DataFrame.
4. Fill Null values and replace 0 stdev values (No dividing by zero!).
5. Calculate measurements for New Days where there is no baseline.
6. Evaluate the maximum risk score by user.
7. Output the behavior for later use.

Employee Flight Risk

Most Insider Threat incidents occur within 90 days of employee departure from the job

How would we identify this type of behavior?

- Visits to Job Search Websites
- Uploads to Job Search Websites
- Email correspondence with competitors

What kind of issues can be expected?

- HR/Recruiters managing job postings
- Hiring Managers actively recruiting new employees



Flight Risk Baselines

```
1 index=web category="Job Search" earliest=-120d latest=-30d  
2 | stats count as num_visits by user, date_mday, date_month  
| stats avg(num_visits) as avg_js_visits by user  
| outputlookup user_flightrisk_baseline.csv
```

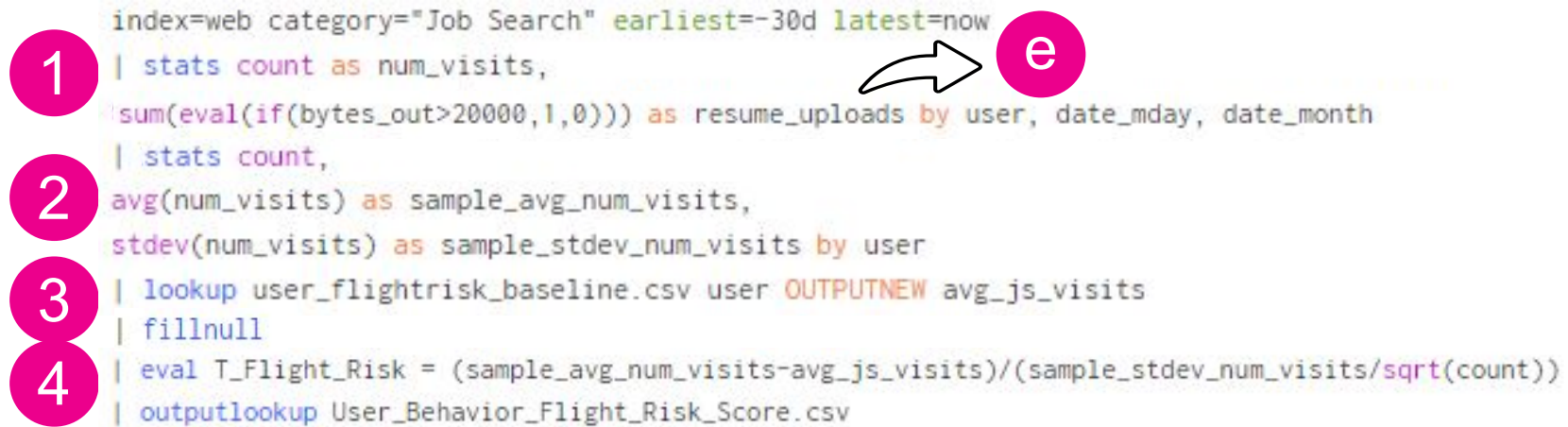
Flight Risk Baselines Search

Let's explain what's going on in that search

1. Calculate daily values for job website views, and then calculate the average per day
2. Output baselines for future use.

Flight Risk Variation

```
1 index=web category="Job Search" earliest=-30d latest=now
  | stats count as num_visits,
  sum(eval(if(bytes_out>20000,1,0))) as resume_uploads by user, date_mday, date_month
2 | stats count,
  avg(num_visits) as sample_avg_num_visits,
  stdev(num_visits) as sample_stdev_num_visits by user
3 | lookup user_flightrisk_baseline.csv user OUTPUTNEW avg_js_visits
  | fillnull
4 | eval T_Flight_Risk = (sample_avg_num_visits-avg_js_visits)/(sample_stdev_num_visits/sqrt(count))
  | outputlookup User_Behavior_Flight_Risk_Score.csv
```



Flight Risk Variation Search

Let's explain what's going on in that search

1. Calculate the count of Job Search site visits by day for users.
 - e. (e for extra credit) This is an example of how to look for uploads such as a resume upload or uploading content to profiles on job sites. Just a demonstration, not actively used here.
2. Calculate the average and standard deviation of the sample.
3. Append the prior dataset to users, fill null values with 0, then calculate risk score for users.
4. Output the behavior to a lookup for later use.

Web Upload Behavior

How do we determine if a web upload is out of the norm?

How would we identify this type of behavior?

- Large one-shot uploads to a site
- Smaller, incremental uploads to a site
- Either of the above to a new/uncommon website

What kind of issues can be expected?

- Uploading content to internal sites such as Sharepoint (Whitelist)
- Depending on measurement, some features such as autosave on Word live documents could be common false positives.



Web Upload Baselines

```
index=web earliest=-120d latest=-30d  
1 | stats sum(bytes_out) as bytes_out by user, website, date_mday, date_month  
  | stats avg(bytes_out) as avg_bytes_out,  
    stdev(bytes_out) as stdev_bytes_out by user, website  
2 | eventstats avg(avg_bytes_out) as total_avg_bytes_out  
  | outputlookup user_webupload_baseline.csv
```

Web Upload Baselines Search

Let's explain what's going on in that search

1. Use stats to measure the total output for users and websites by day. Calculate descriptive statistics.
2. Calculate the average of the average output for user to fill nulls later. Then output results to a lookup for later use.

Web Upload Variation

```
index=web earliest=-30d latest=now
| stats sum(bytes_out) as bytes_out
1 | by user, website, date_mday, date_month
  | stats count, avg(bytes_out) as sample_avg,
  stdev(bytes_out) as sample_stdev,
  sum(bytes_out) as total_bytes_out by user, website
  | lookup user_webupload_baseline.csv user, website
2 | OUTPUTNEW avg_bytes_out, stdev_bytes_out, total_avg_bytes_out
  | eventstats avg(stdev_bytes_out) as total_stdev_bytes_out
  | eval avg_bytes_out = COALESCE(avg_bytes_out, total_avg_bytes_out),
3 | sample_stdev=COALESCE(stdev_bytes_out, total_stdev_bytes_out)
  | eval T_Web_Upload = (sample_avg-avg_bytes_out)/(sample_stdev/sqrt(count))
```

Web Upload Variation Search

Let's explain what's going on in that search

1. Use stats to measure the total output for users and websites by day. Calculate descriptive statistics including count.
2. Use lookup from previous step to add values, take the average of the sample standard deviation to fill null values.
3. Fill nulls, then calculate the Web Upload Risk Score



Learning From the Data

Let's use MLTK!

Many Dimensions of Behavior

Multiple Measurements of User Behavior

- User Login Variance
- User Flight Risk Score
- User Variance in Web Uploads

Identifying Outliers

- Primary measurement: Web Upload Variance
- Secondary, Enriching Values: Flight Risk Score, Login Variance

“To deal with a 14-dimensional space, visualize a 3-D space and say ‘fourteen’ to yourself very loudly. Everyone does it.” – Geoffrey Hinton

DBSCAN Algorithm

This awesome algorithm ships with Splunk MLTK

Density Based Spatial Clustering of Applications with Noise

- MLTK uses scikit-learn library for DBSCAN
- What is DBSCAN composed of?
 - Core Points
 - Border Points
 - Noise Points
- Parameters/Tuning
 - Minimum Family Size of 5 (auto: min_samples=5)
 - Epsilon



DBSCAN is great at separating clusters of high-density and at identifying noise from those clusters.

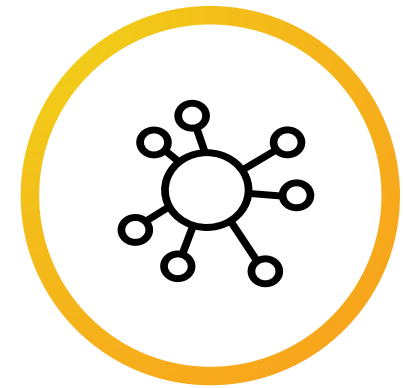
Unsupervised Learning

Now that we've done the fun part of aggregating data, let's get some results!

Identifying Irregular Outbound Data

- Start with search for primary activity
- Append user activity with behavior risk scores
- Identify outliers with algorithm
- Filter out noise

```
| lookup User_Risk_Scores.csv user OUTPUTNEW T_Flight_Risk, Z_Login_Variance  
| eval T_Web_Upload=max(0,T_Web_Upload),  
T_Flight_Risk=max(0,T_Flight_Risk)  
| fit DBSCAN eps=0.3 T_Web_Upload, T_Flight_Risk, Z_Login_Variance  
| search total_bytes_out>1024*1024, cluster=-1
```





Integrating with ES

We've got good results, now let's tell the SOC

Connecting with the SOC

Solutions for Splunk ES as well as Core Splunk

Now that the results have been generated, how do we get it in front of the SOC Analysts?

- Notable Events/Splunk Alerts to view a Dashboard
- Notable Events/Splunk Alerts with content from lookups

What content should we display?

- Drill-Downs from Dashboards for SOC Analysts to dig deeper
- Enriching Details on Dashboards/Lookups
 - `| eval Comment1 = if(Risk_Score > 7,"User is a big risk, watch out!", "This user is chill, bro.")`
 - `| eval Comment2 = if(total_bytes_out>1024*1024,"User uploaded ".total_bytes_out." to ".website, "")`
 - `| eval User_Behavior = Comment1 + " " + Comment2`

Dashboard

What are the benefits

Alert a SOC Analyst with “Check this Dashboard”

- Set to run on a schedule
- Send an alert/notable event

Dashboard can provide additional detail

- Comments on User Behavior from a variety of sources (Login Time/Flight Risk Score)
- Drill-Downs with tokens can show additional detail
 - What day the user logged in at an odd time
 - What websites and times a user used Job Search sites
 - Timestamps, bytes_out, and full url's for web uploads to the site in question



Lookups

A more advanced method

Alert a SOC Analyst with New Results

- Output a lookup with positive results on a schedule
 - When outputting results, include a time value (`| eval time = now()`)
 - Use `inputlookup` to pull in results

Sort by time

Use `streamstats count` and `last(bytes_out)`

Use `eventstats` to identify most recent wave of results

By identifying the most recent events, and the variance between prior events and the current one – filtering can be done to alert the SOC only when there is a New Event or when a user who previously was alerted on has now sent much more outbound data.



Key Takeaways

Detecting and Mitigating Insider Threats Using MLTK and Splunk Enterprise Security

1. Insider Threats can be costly to organizations, but Splunk MLTK and ES can be used to quickly understand and react.
2. Splunk MLTK can be used to identify outliers quickly and also enrich alerts for SOC Analysts in a way they can understand.
3. Integrating Splunk MLTK and Splunk ES is an essential part of building an effective insider threat program and there's multiple ways to do it. So let your SOC know what you've found!



splunk>

Thank

You



Go to the .conf19 mobile app to

RATE THIS SESSION

