Detection Technique Deep Dive

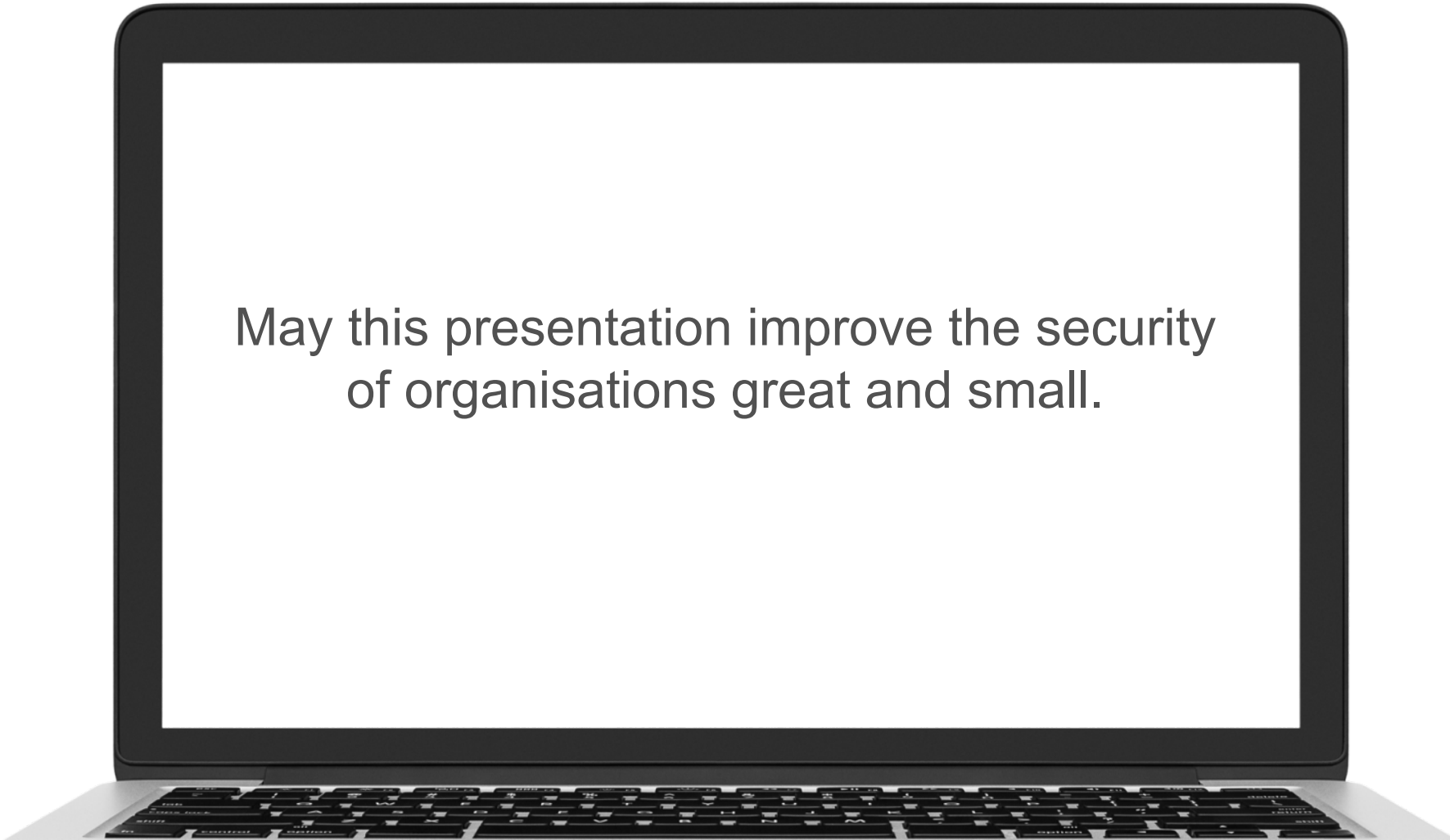
Doug Brown | Senior Information Security Analyst, Red Hat
95B6 922E 47D2 7BC3 D1AF F62C 82BC 992E 7CDD 63B6

October 2018 | Orlando, United States

Forward-Looking Statements

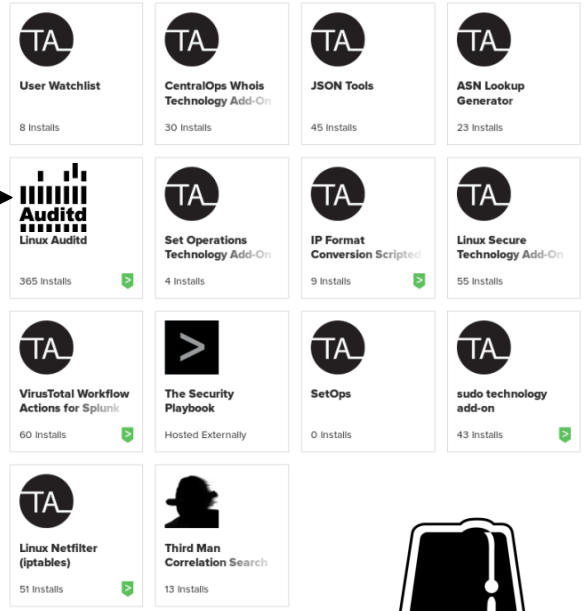
During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. 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 functionality described or to include any such feature or functionality in a future release.



Speaker Background

- Doug “trustedsubject” Brown
- Fond of SELinux and elegant SPL
- SplunkTrust member
- Author of more than a dozen Splunkbase apps, incl Auditd
- 2016 Developer Revolution Award Winner
- Masters degree examining the compositional behavioural properties of computer networks using formal methods: https://eprints.qut.edu.au/93693/1/Douglas_Brown_Thesis.pdf
- Contributor to ES roadmap
- Preparing for a Successful ES Engagement: https://www.splunk.com/blog/2016/10/24/preparing-for-a-successful-enterprise-security-ps-engagement.html



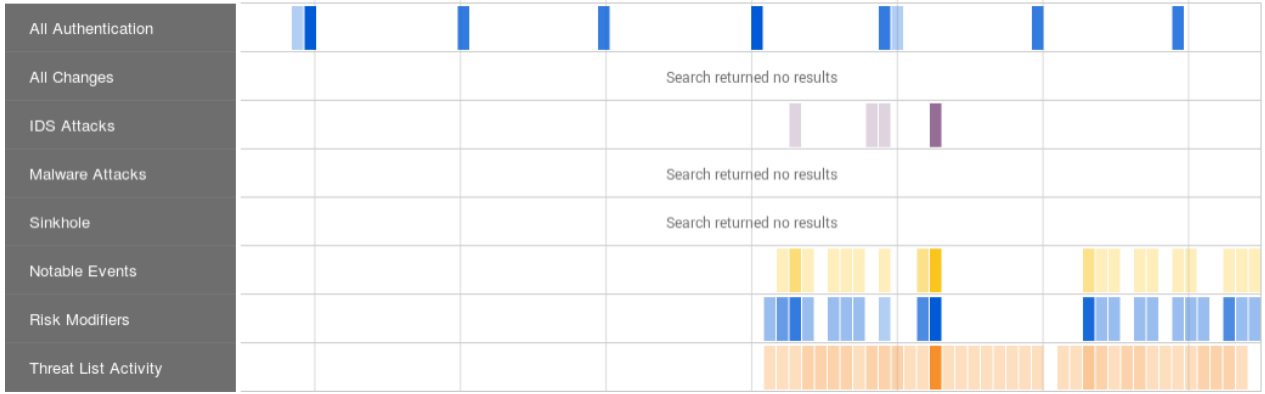
SPLUNK TRUST

splunk> .conf18

redhat Operational Security

- Leading Open Source vendor
- Relatively small global team
- Splunk customer since 2012
- Multi TB license

Splunk Enterprise Security™



Outline

- Introduction
- Numerical Techniques
- Categorical Techniques
- Appendix
- Additional Resources

Target Audience

This session is intended to be:

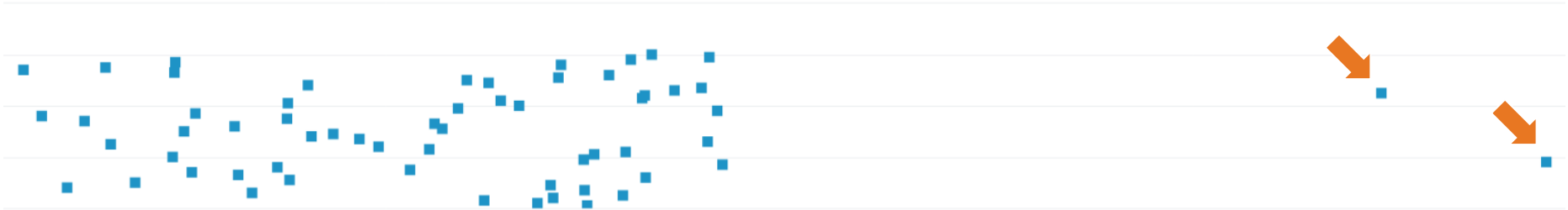
- A master class on SPL detection techniques
- Framed in a security context, but with universal techniques
- An advanced session in terms of sophistication, but not complex/difficult

Background

This session addresses how to detect changes in behaviour in the third/fourth stage (Metric/Conditions) of the *Correlation Search Development Process*:

- Brown, D. (2017). *The Art of Detection Using Splunk Enterprise Security* [Video Recording]. Retrieved from: https://conf.splunk.com/sessions/2017-sessions.html#search=The%20Art%20of%20Detection%20Using%20Splunk%20Enterprise%20Security&

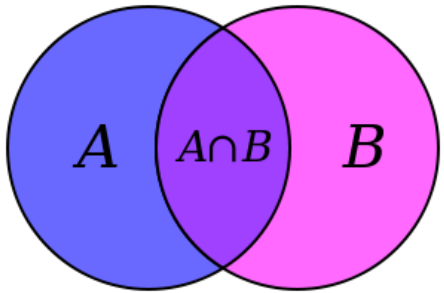
Problem



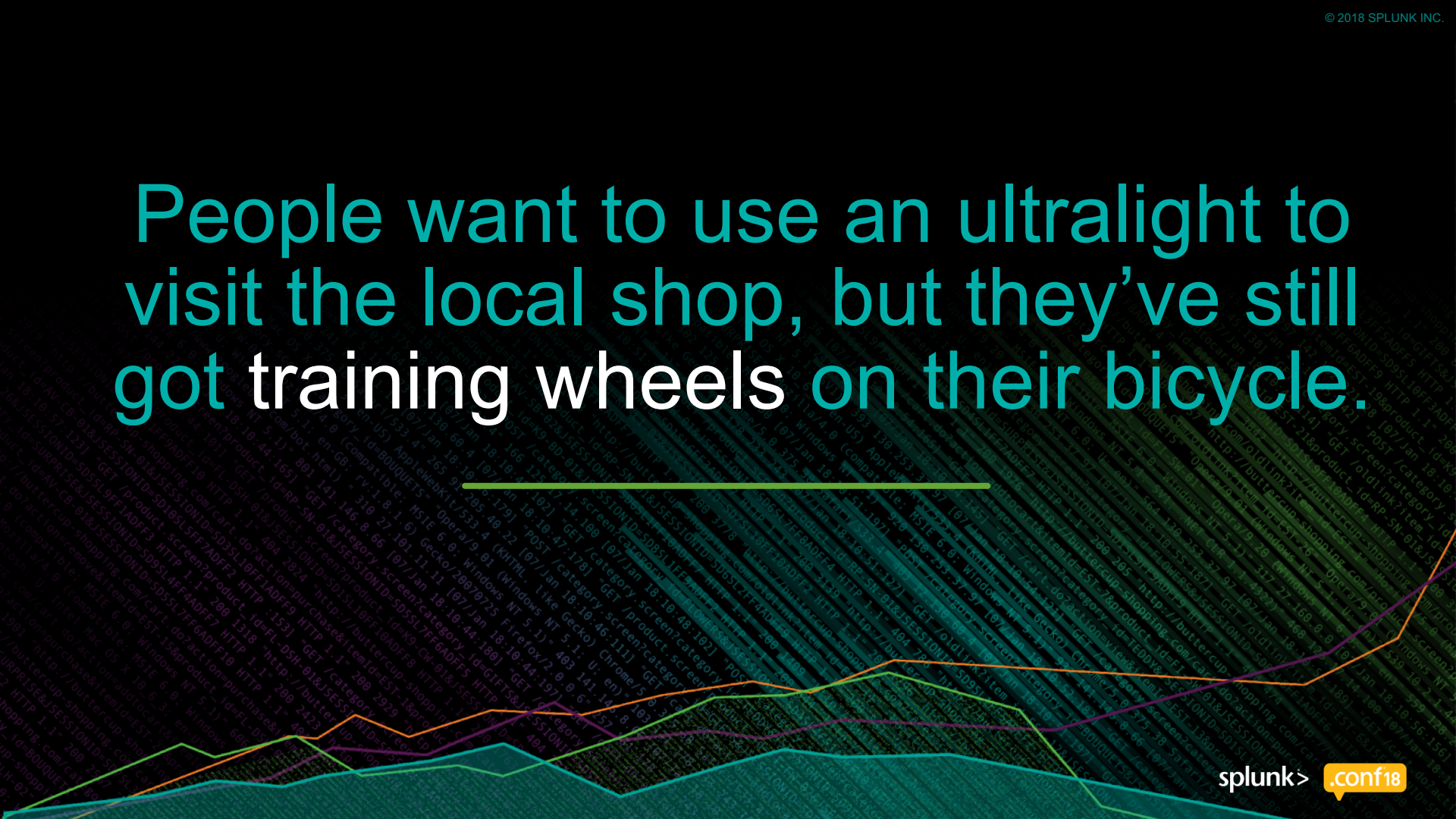
We want Splunk to “see” like humans (or better).

Goal

Produce a set of canonical techniques for finding anomalies and codify their use.



People want to use an ultralight to visit the local shop, but they've still got training wheels on their bicycle.



Scope

- **Change-based** outlier anomalies
- Numerical techniques only consider **increases**
- Only seasonality that is evident from _time
- Event sequence changes are out of scope
- Frequency-based detection is out of scope

  (e.g. timing covert-channels)

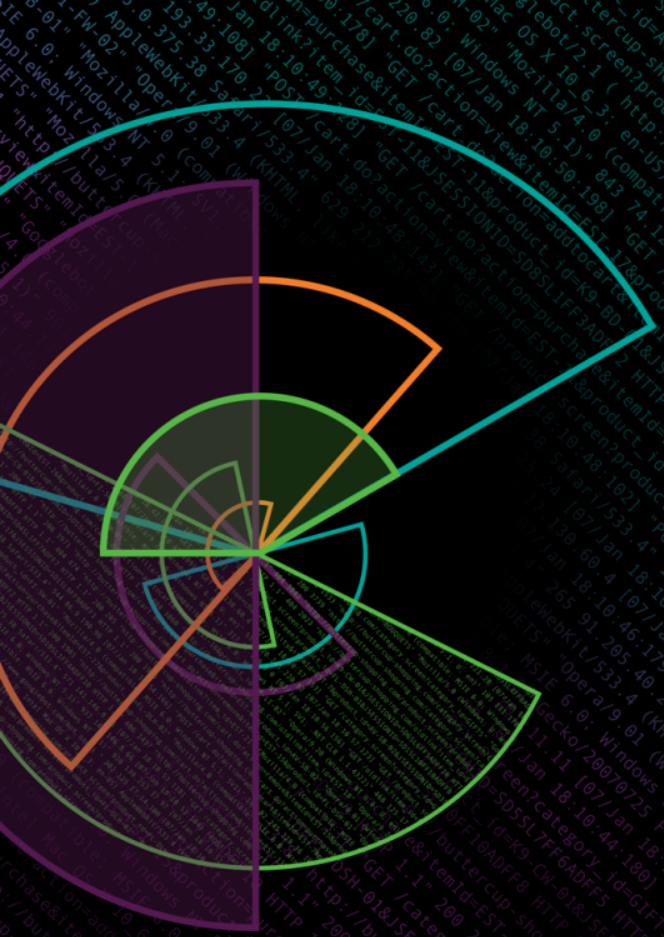
Methodology

1. Enumerate different types of data and their anomalies
2. Discuss progressively more complex techniques and their limitations
3. Demonstrate how techniques work with agnostic case studies

- Searches are provided in detection rather than hunting/drill-down form (e.g. … | where y>avg+2*stdev instead of … | eval k=(y-avg)/stdev)
- In the slides, ***y*** is the name of the field you're looking for anomalies within
- Stat commands can use ***by*** to support multiple independent variables
- [tm]stats is best (we assume its use as the generating command here)

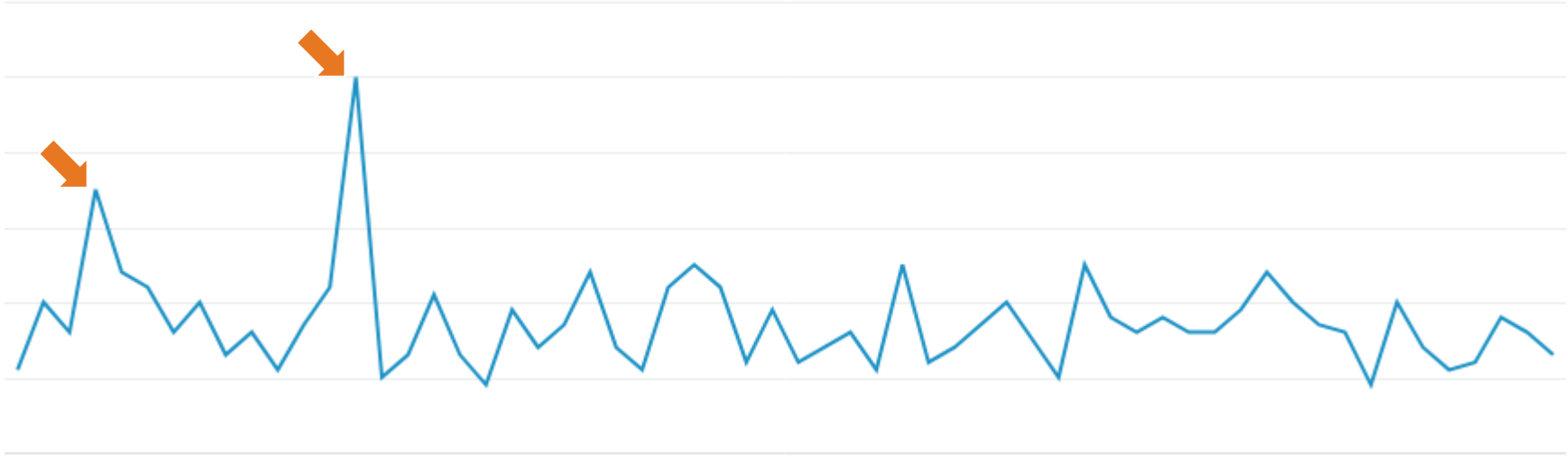
Axioms

- This is **not** an exhaustive list of techniques
- There's more than one way to skin a cat
- No technique is purr-fect
- All techniques are subject to continuous improvement
- Many variations on these core techniques exist
- There's no **"wrong"** technique, but appropriateness/efficacy varies



Numerical Techniques

First Case Study: Linear With Low Variability



1. Global Average

Sometimes a simple average is all that's needed and is easy to understand:

… | eventstats avg(y) as avg

| where y>#*avg

Where # is a number such as 2 (i.e. double the average)

Limitations:

- "Supervised" calibration (requires analyst to manually determine #)
- Doesn't work well for data with "seasonality"
- Doesn't work well for highly variable datasets
- Doesn't work well where trend isn't flat

Second Case Study: Linear With High Variability

2. Standard Deviation

The technique often used with (1) to take into account variability:

… | eventstats avg(y) as avg, stdev(y) as stdev

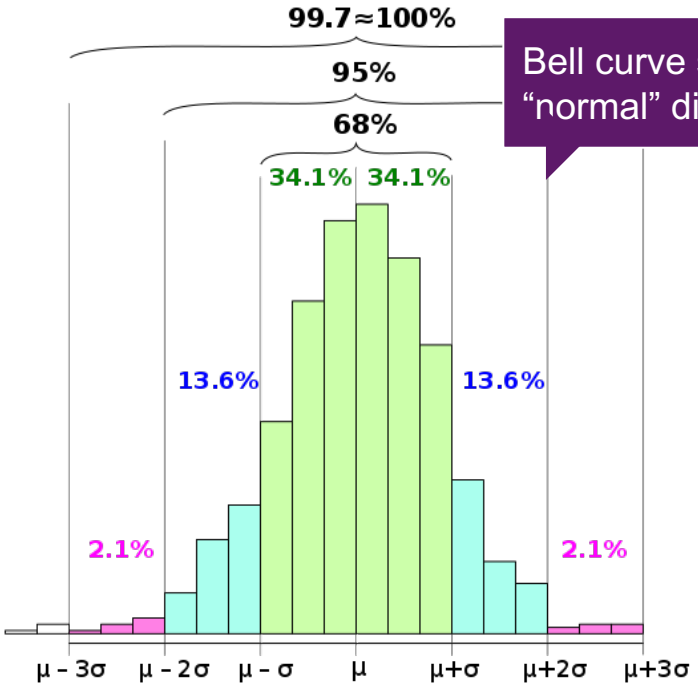
| where y>avg+#*stdev

According to *Chebyshev's inequality (Нера́венство Чебышёва)*, a **#** value of 2~10 should find unusual/anomalous values in **any** distribution.

Limitations:

- Same limitations as (1), except high variability

The Issue of Distribution



Bell curve shape of a “normal” distribution

Patchy distribution shape we often see in IT

https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule#/media/File:Empirical_rule_histogram.svg

splunk> .conf18

3. Moving Average

Compare datapoint to average *so far*, instead of average across whole dataset:

… | streamstats current=f avg(y) as avg

| where y>#*avg

Limitations:

- Same limitations as (1), but changing trends can be accommodated using “window” and “global” arguments to streamstats.

4. Large Increase

Find a percentage increase (relative to the previous value):

… | streamstats current=f last(y) as last

| where y>#*last

A # value of 2 would trigger if y doubled between one value and the next.

Limitations:

- Same limitations as (1)
- Prone to false-positives when y recovers to normal range.

5. Differential Calculus

The “rate of change” (relative to the previous value):

… | streamstats current=f last(y) as last

| where y-last>#

Used in cases where you’re concerned about y increasing by a known discrete amount #, rather than a relative proportion such as in (3).

Limitations:

- Similar limitations as (4)

Discussion

- (1), (2) and (3) measure the size of a datapoint in contrast to (4) and (5), which measure the *increase* in size.
- Generally we only want to know if something is both big in the context of the dataset **and** increases dramatically relative to the previous data points.
- For this reason, we often combine both types of techniques to find changes of genuine interest.

- Be aware that your visual perception of change can be fooled by graph axis - the machine sometimes knows best.

6. Combining Multiple Technique Types

Supports multiple independent variables

Suggested combination for very large non-seasonal datasets:

```
… | streamstats current=f count as datapoint, avg(y) as avg_y, stdev(y) as stdev_y, last(y) as last by field1, …
  | eval change=y/last
  | streamstats current=f avg(change) as avg_change, stdev(change) as stdev_change by field1, ...
  | where datapoint># AND y># AND change>avg_change+#*stdev_change AND y>avg_y+#*stdev_y
```

Minimum number of previous events for baselining

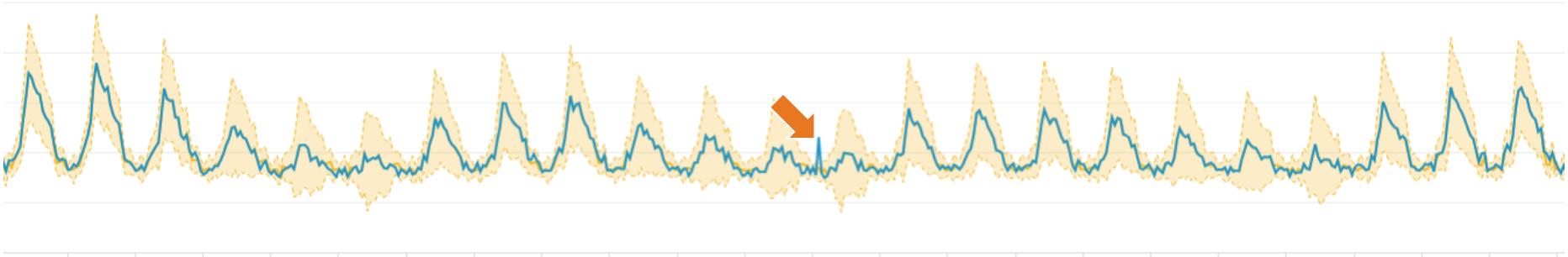
Known minimum value of interest and prevents triggering under unusual circumstances where y recovers

Techniques 2 & 4

Techniques 2 & 3

splunk> .conf18

Third Case Study: Seasonal Data



7. `Brown Manoeuvre()`

Preferred technique for automatically finding anomalies in seasonal datasets:

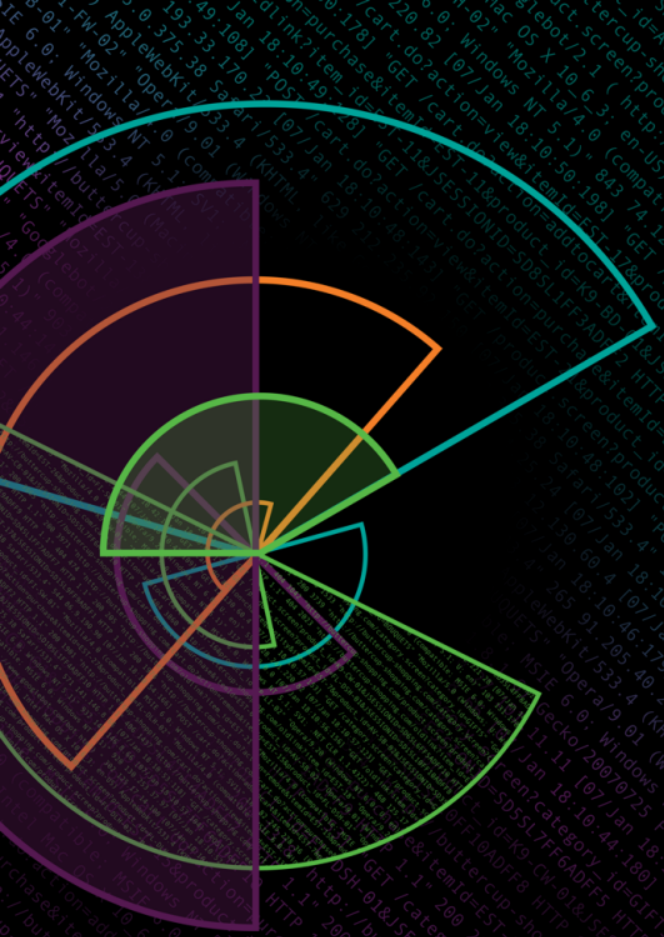
```
… | predict y future_timespan=0 [algorithm=LLP]
    | rename lower95(prediction(y)) as lower, upper95(prediction(y)) as upper
    | eval range=upper-lower
    | eval difference=case(y>lower AND y<upper, 0, y<lower, round((y-lower)/range,1), y>upper,
      round((y-upper)/range,1))
    | where difference>#
```

Any *difference* value greater than 0 is an anomaly so **#** is generally set between 0 and 1, depending on how anomalous y has to be before you are concerned.

Less “supervised” than (6) but heavier, limited and more black box (uses *Kalman Filter*).

Limitations:

- Only one numerical dependant variable
- Only temporal (time-based) datasets
- Only seasonality that is evident from time



Categorical Techniques

Fourth Case Study: Categorical Anomalies

field1	field2	field3
A	B	C
A	B	D
A	B	C
B	B	D
B	E	D
B	B	D

Completely distinct rows have not been marked.

A. Unique Field Value

When a new value is seen in a field:

… | eventstats count by fieldA

| where count==1

Limitations:

- Baselining threshold needs to built into where statement or alert conditions e.g. *_time>relative_time(now(),"-1h")*
- In some cases, a higher threshold (such as **<3**) may be necessary to prevent false-negatives. Consider finding rare rather than unique values using:

  … | streamstats count as datapoint | streamstats count by fieldA | where count/datapoint<0.01

B. Unique Tuple

When the combination of values from multiple fields in an event is unique:

… | eventstats count by fieldA, fieldB, … fieldn

| where count==1

Limitations:

- Same limitations as (A)
- Doesn't indicate which fields contain unique values
- Doesn't indicate the extent to which the event's field values are different

C. Distinct Fields

Identify an event's fields with unique values compared to the whole dataset:

… | distinctfields [by=fieldZ] fieldA fieldB ... fieldn

| eval mvcount=mvcount(distinctfields)

| where mvcount>#

will be between 1 and the number of fields provided to distinctfields command.

Limitations:

- The distinctfields command (https://splunkbase.splunk.com/app/3516/) is not a reporting command (retains full events), so filter-in only the necessary fields (e.g. using the table command before) to improve performance.

C. Distinct Fields

Correlation Search Example

New Distinct SELinux AVC Tuple

```
| tstats summariesonly=t count from datamodel=Auditd where (nodename = Auditd.AVC)
groupby _time, host, Auditd.scontext_domain, Auditd.tclass, Auditd.perm, Auditd.tcontext_type span=1d
| `drop_dm_object_name("Auditd")`
| distinctfields by=scontext_domain tclass perm tcontext_type
| where mvcount(distinctfields)>1 AND _time>relative_time(now(),“-1d”)
```

Auditd

Data model provided by Linux Auditd app (https://splunkbase.splunk.com/app/2642/).

splunk> .conf18

D. Increase in Distinct Fields

This **example** shows how categorical and numerical techniques can be combined to detect an *unusual increase* in categorical anomalies by something.

```
… | distinctfields by=fieldZ fieldA … fieldn
    | eval mvcount=mvcount(distinctfields)
    | bin span=1h _time AS hour
    | stats sum(mvcount) as sum by hour, fieldZ
    | streamstats current=f avg(sum) as avg, stdev(sum) as stdev by fieldZ
    | where sum>avg+#*stdev
```

Hunting Example with BOTSv1

```
index=botsv1 sourcetype=XmlWinEventLog:Microsoft-Windows-Sysmon/Operational CommandLine=*
| eventstats count by host, CommandLine
| timechart span=1h sum(eval(count==1)) as sum
| eventstats avg(sum) as avg, stdev(sum) as stdev
| eval k=(sum-avg)/stdev
| sort - k
```

All time

✓ 32 events (before 7/19/18 1:46:00.000 AM) No Event Sampling Job Smart Mode

Events Patterns Statistics (334) Visualization

100 Per Page Format Preview

< Prev 1 2 3 4 Next >

_time	sum	avg	k	stdev
2016-08-24 16:00	128	51.166666666666664	1.559102471907961	49.28048971618146
2016-08-10 22:00	91	51.166666666666664	0.8082982446551031	49.28048971618146
2016-08-10 21:00	50	51.166666666666664	-0.02367400716563059	49.28048971618146
2016-08-24 17:00	19	51.166666666666664	-0.6527261975666732	49.28048971618146
2016-08-24 18:00	18	51.166666666666664	-0.6730182037086423	49.28048971618146

live.splunk.com/splunk-security-dataset-project

splunk> .conf18

Summary

Techniques Provided

Numerical:

- Global Average
- Standard Deviation
- Moving Average
- Large Increase
- Differential Calculus
- Brown Manoeuvre

Categorical:

- Unique Field Value
- Unique Tuples
- Distinct Fields
- Increase in Distinct Fields

Key Takeaways

1. Change-based detection doesn't require complex mathematics
2. Only a small number of reusable techniques are required
3. Combine techniques for greater efficacy

Appendix

Hypothetical Example

Using Correlation Search Development Process

Hypothetical Example Using Correlation Search Development Process

Stage 1 (The Idea):

- We observe with a past incident an increase in threat list activity via the Asset Investigator
- We would like to know when a significant increase occurs for a device in the future
- We find that the number of “threat keys” hit by a src is particularly indicative of concern

Stage 2 (The Source):

- ES automatically correlates and summarises network traffic that hits threat intel
- tstats against datamodel counting the number of unique threat keys each src hits per hour:

```
| tstats `summariesonly` dc(Threat_Activity.threat_key) as count from datamodel=Threat_Intelligence
where nodename=Threat_Activity AND Threat_Activity.src=10.* groupby  _time, Threat_Activity.src span=1h
| `drop_dm_object_name("Threat_Activity")`
```

Hypothetical Example Using Correlation Search Development Process

Stage 3 (The Metric):

- We need to baseline the typical threat list key count of each src so a numerical technique is required
- Because there's multiple independent variables (i.e. src), technique (6) is a better fit than (7)

```
… | streamstats current=f avg(count) as avg_count, stdev(count) as stdev_count, last(count) as last_count
    by src
  | eval change=count/last_count
  | streamstats current=f avg(change) as avg_change, stdev(change) as stdev_change
```

Hypothetical Example Using Correlation Search Development Process

Stage 4 (The Conditions):

- Hosts seem to often increase up to a count of 3 different threat list keys, so we decide to have more than 3 hits as a minimum (count>3)
- >3 stdev finds the changes in behaviour at the fidelity the team required

```
… | fillnull value=0
  | where count>3 AND change>avg_change+3*stdev_change AND count>avg_count+3*stdev_count
```

Hypothetical Example Using Correlation Search Development Process

Stage 5 (The Triage):

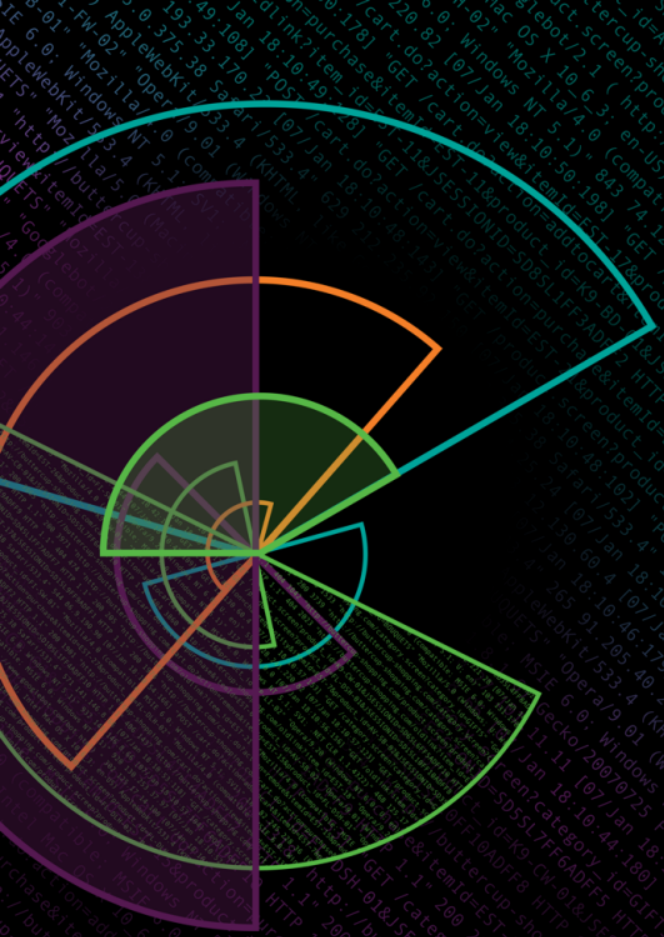
- The src value will always be internal so it's safe to attempt a reverse resolution, adding enrichment for the analyst (could also be used to filter out DNS servers and other known FPs)
- We dynamically calculate the risk aggregated against the src according to how many threat list keys they hit (scaled from 30+ taking into account the minimum count from Stage 3)

```
… | lookup dnslookup clientip AS src OUTPUT clienthost AS src_host
  | eval risk_object=src, risk_object_type="system", risk_score=(count*10)-20
```

https://docs.splunk.com/Documentation/ES/latest/User/RiskScoring#Assign_risk_through_a_search

“ Knowledge is of no value unless you put it into practice.”

Anton Chekhov



Q&A

splunk> .conf18

Thank You

Don't forget to rate this session in the .conf18 mobile app

Additional Resources

Eval / Where Functions:

- docs.splunk.com/Documentation/Splunk/latest/SearchReference/CommonEvalFunctions

Statistical Functions:

- docs.splunk.com/Documentation/Splunk/latest/SearchReference/CommonStatsFunctions

Standard Deviation:

- https://www.khanacademy.org/math/probability/data-distributions-a1/summarizing-spread-distributions/a/introduction-to-standard-deviation

Predict Command:

- docs.splunk.com/Documentation/Splunk/latest/SearchReference/Predict

tstats Command:

- Please see final bonus slide
- conf.splunk.com/sessions/2016-sessions.html#search=_raw%20to%20tstats

Bonus: Median Absolute Deviation

Alternative to Standard Deviation:

```
… | eventstats median(y) as median
    | eval ad=abs(y-median)
    | eventstats median(ad) as mad
    | where y>median+mad*3
```

Consider using this technique if the # value required for Standard Deviation is high (can be caused by large outliers skewing the mean).

Bonus: Interquartile Range

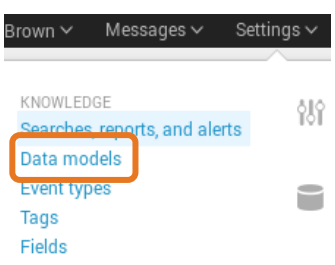
One way to compare an object's behaviour to others (with other techniques):

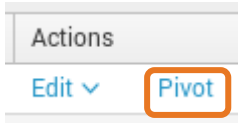
```
… | eventstats median(y) as median, perc25(y) as Q1, perc75(y) as Q3
    | where y>median+1.5*(Q3-Q1)
```

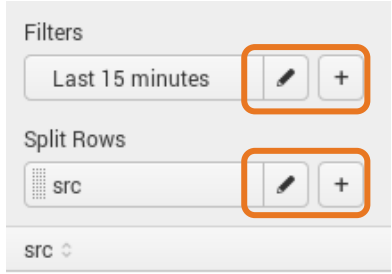
The object may be behaving unusually, but adding a comparison to the whole dataset gives context that might be useful in some anomaly detection use cases.

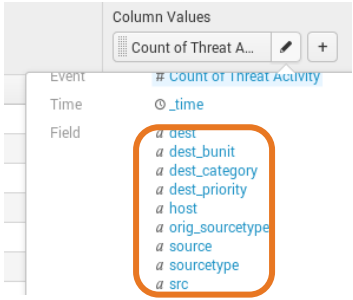
Thanks **George Starcher** for his feedback to add this material.

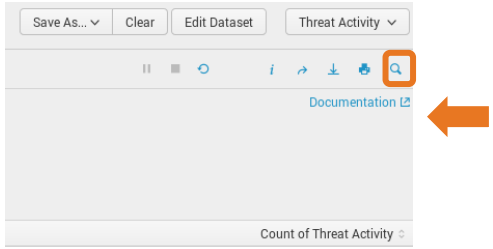
Data Model tstats via GUI

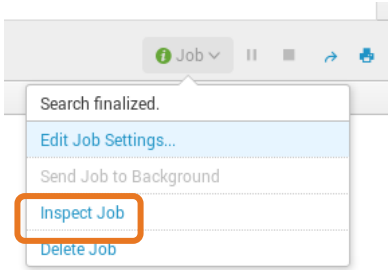












eventIsStreaming	true
eventIsTruncated	true
eventSearch	tstats count AS "Count of Threat Activity" from datamodel=Threat_Intelligence where (nodename = Threat_Activity) groupby Threat_Activity.src prestats=true
eventSorting	none

1. Settings - > Data models
2. Pivot on Data model
3. Select time period and add filters (e.g. src=10.*)
4. Select stats required
5. Click magnifying glass
6. Inspect Job
7. Find eventSearch
8. Copy tstats command without prestats=true

When using tstats, add summariesonly=t option or `summariesonly` macro in ES search environment.

Data model node name can be removed from all field names by piping tstats to: |`drop_dm_object_name("Node_Name")`

splunk> .conf18