splunk> .conf2017

# Security Ninjutsu Part Four

## The SPLening

## 2.5 hours of EPIC SPL stuffed into 45 minutes

David Veuve | Principal Security Strategist

September 2017 | Washington, DC

# 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.



splunk> .conf2017

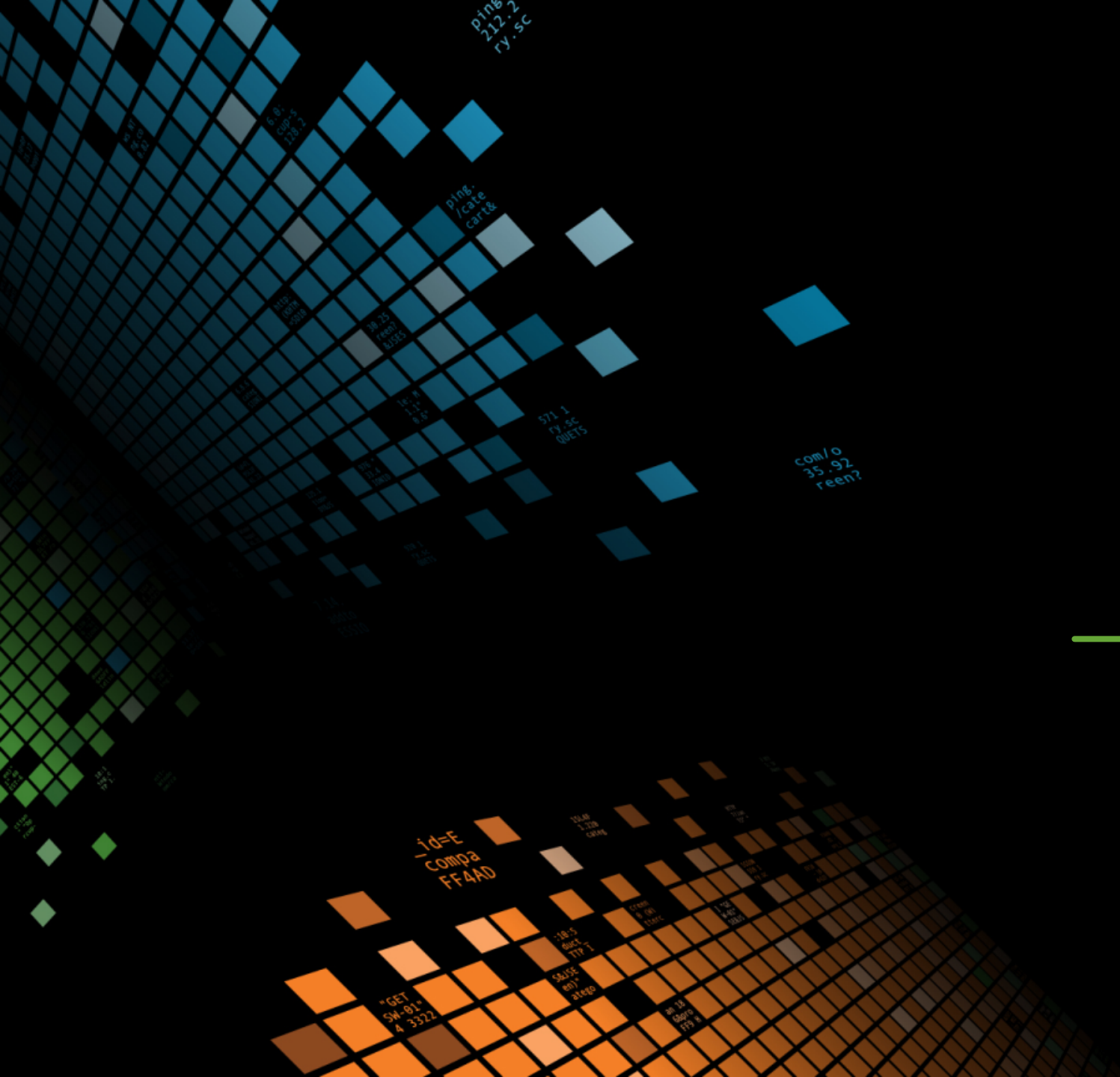
# Forward Looking Errata

- Having just completed the first draft of this presentation, I can guarantee you that there will be updates. Check out https://dvsplunk.com/ for those updates!

# Personal Introduction

- **David Veuve**
  Principal Security Strategist, Splunk
- SME for UEBA, Security, Architecture
- dveuve@splunk.com
- Former Splunk Customer
- Primary author of the Splunk Security Essentials app

- **2017 Talks:**
  - Security Ninjutsu Part Four (Hi!)
  - Searching FAST: Start Using tstats and other acceleration techniques
  - Quickly Advance Your Security Posture with Splunk Security Essentials
- **Prior Conf Talks:**
  - How to Scale Search from _raw to tstats
  - Security Ninjutsu Part Three: .conf2016
  - Security Ninjutsu Part Two: .conf 2015
  - Security Ninjutsu Part One: .conf 2014
  - Passwords are for Chumps: .conf 2014



# Intro

Section subtitle goes here

splunk> .conf2017

# Past Security Ninjutsus

## Part One: 2014

- Visibility, Analysis, *AND* Action
- David's First Anomaly Detection

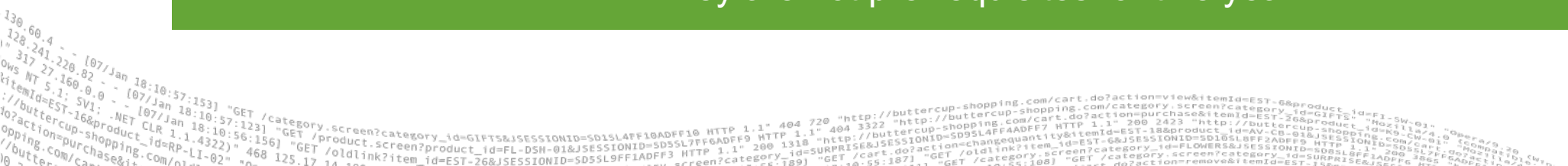
## Part Two: 2015

- Correlation Across Multiple Sourcetypes
- Risk Across The Org.. In Splunk!
- Strategies to Counter Alert Fatigue

## Part Three: 2016

- Real Correlation Searches from Real Customer
- Content Development Process

There is lots of valuable content in the prior Ninjutsus - I highly recommend you visit them. They are not pre-requisites for thie year.



splunk> .conf2017

# Ninjutsu 2017!

## SPL AWESOMESAUCE

- If you forced my co-workers to pick a single top skill of David Veuve, single top reason to bring him into a meeting, it is: SPL Skills
- Let David be David - here is all the SPL fit to print
- You are looking at the PDF copy of these slides. There is a lot of context and explanation in these slides.
- I recommend checking out the video as well, to help reinforce the key takeaways, and use this PDF as the reference to implement those ideas

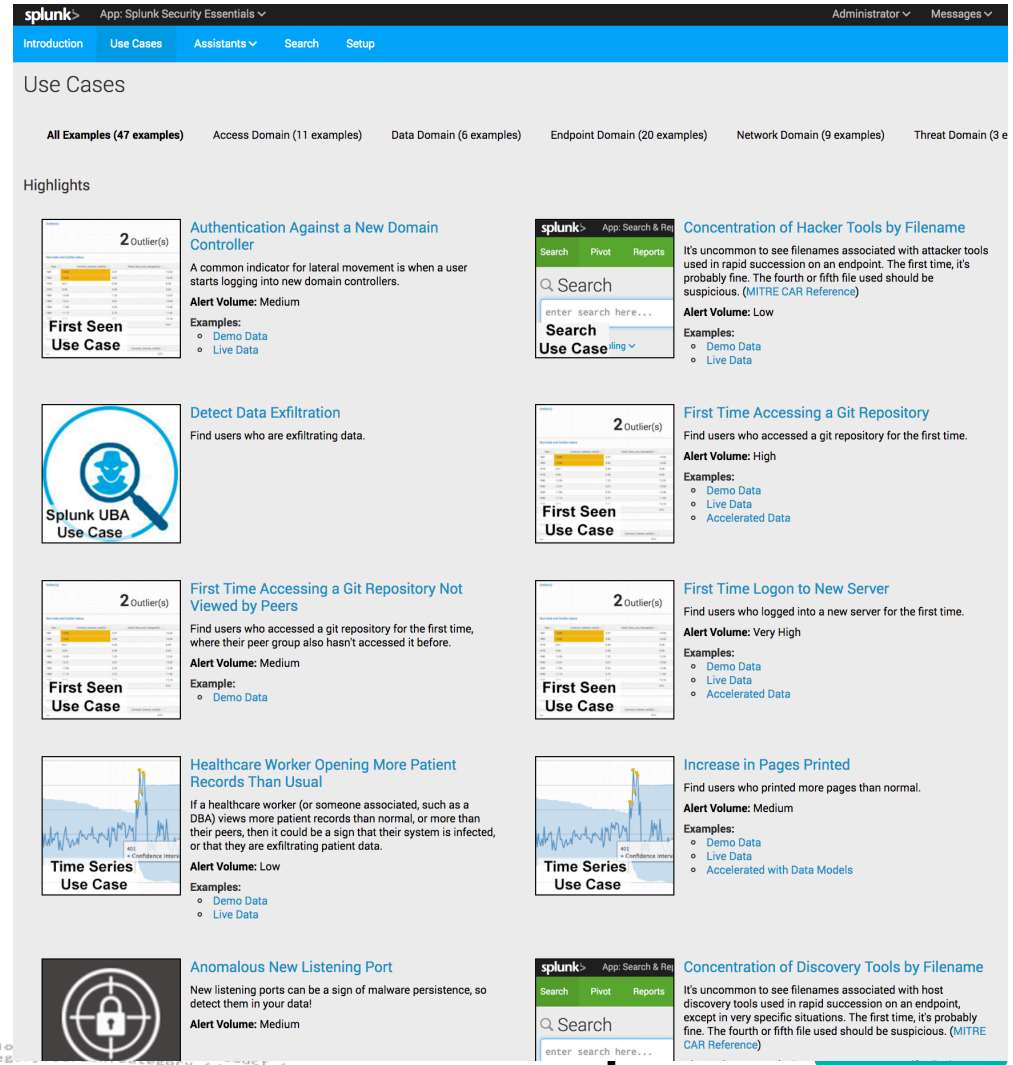
# Oh Snap, there's an App?

What's the happs.. there's an app?

## Splunk Security Essentials

https://splunkbase.splunk.com/app/3435/

- Not explicitly focused on this session, but lots of good working detection logic
- Also demonstrates what you *can* do with Splunk and Security Detection



# So What Are We Going to Talk About?

Did someone say "Obligatory Word Cloud"?

action advanced alert analysis analysts app approach avg background build capability case challenges checking clustering com commands conf count customers data days dest detection different etc eval events example feb fields general host http id index info ip learning logic logs lookup machine malware mid ml name number numeric per proxy ratio results rid risk scenarios search security series server sessions soc sources sourcetype spl splunk src stats stdev summary table technique timestamps transaction tstats url user value wed working

splunk> .conf2017

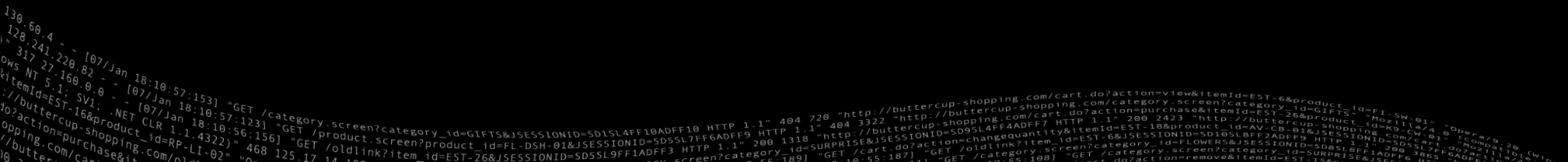
# Most Important Announcement

- Very little of what was covered in this deck was actually built or discovered by me.
- We all stand on the shoulders of giants, and I like to think the giants I stand on the shoulders of are as tall as they come.
- Major shout outs go to far too many people to mention here, including many of my customers who have come up with innovative ideas.
- Also, you know, the engineers deserve some credit for building the product in the first place, and anticipating so many needs that we would have while also allowing it to support so many needs they couldn’t anticipate.

- /me hat tips

# Let's Get Into It

- Over the rest of the presentation, we will go through different SPL techniques that have opened eyes and helped Splunkers in the past.
- For each technique we will state:
  - Problem statement: Why do you care?
  - Describe the technique: How do you solve?
  - Lots of real SPL to do this: Pics or it didn't happen



# Intermediate Techniques

Where was introductory? This presentation starts at 5 and goes to 15.

splunk> .conf2017

# Technique: Common Information Model

## Background and Challenges

- "I have 7 different sourcetypes with different field names for authentication and I want to write just one search that crosses all of those"
- "I want to write a presentation on security SPL in a way that can apply to any security customer and represent concepts in generic terms"

- The common information model allows us to do exactly that - use a single nomenclature across many searches, many sourcetypes, in many environments

splunk> .conf2017

# Technique: Common Information Model

- Building a dashboard? Print something good, and then drilldown well.

```
tag=authentication

| chart count over src by action

| where success>0 AND failure>10
```

We can just say tag=authentication instead of specifying our Windows Logs, Linux Logs, PAN Auth Logs, Oracle Auth Logs… (I could go on all day)

action is a field defined in the Common Information Model - look at how we can just reference it so easily and it will track successes or failures across all sourcetypes

Ah, building correlation searches on Splunk is easy!

# Technique: Common Information Model

## Resources

- Conf 2016: The Power of Data Normalization: A Look at CIM Under the Hood
  Mark Bonsack and Vlad Skoryk
  - http://conf.splunk.com/files/2016/slides/the-power-of-data-normalization-a-look-at-cim-under-the-hood.pdf
  - http://conf.splunk.com/files/2016/recordings/the-power-of-data-normalization-a-look-at-cim-under-the-hood.mp4

# Technique: Eval

## Background and Challenges

- I assume you already know the basics of eval, but there are several functions I use often that have wow'd some people.
- "How do I deal with different data types containing multiple field names?"
- "I hate nested if statements!"
- "I want to do super advanced string manipulations"

- Even if you don't see an immediate use case for these techniques, remember them. I promise it will be worth it.

# Technique: Eval

## The coalesce Function

- coalesce is an often overlooked function, that will return the first non-null value.
- Whenever you have multiple data types, you will invariably have different field names for the same value. Combining them into one field without overwriting different values is done by most Splunk users with if statements or other extreme hijinks. coalesce is much easier.

```
(sourcetype=datasource1 Source_IP=*) OR
(sourcetype=datasource2 srcip=*)

| eval src_ip = coalesce(srcip, Source_IP)
```

Begin with your disparate data sources

For datasource1, srcip will be null. For datasource2, Source_IP will be null. src_ip will always have the right result.

splunk> .conf2017

# Technique: Eval

## The case Function

- case is skipped by almost everyone who has never been a programmer. If you've been a programmer, you already know about it. If you haven't, get psyched.
- One of the greatest strengths of eval is that it allows you to embed all manner of business logic. Invariably, this requires you to have if statements… but often, you end up with multiple scenarios. If a, then x, if b, then y, if c, then z, if d, then throw an error.
- Many use nested if statements, but case handles multiple conditions with ease.

```
sourcetype=datasource1

| eval direction = case(
   cidrmatch("10.0.0.0/8", src_ip) AND NOT
            cidrmatch("10.0.0.0/8", dest_ip), "outgoing",
    NOT cidrmatch("10.0.0.0/8", src_ip) AND
            cidrmatch("10.0.0.0/8", dest_ip), "incoming",
    cidrmatch("10.0.0.0/8", src_ip) AND
            cidrmatch("10.0.0.0/8", dest_ip), "internal",
    1=1, "outgoing to outgoing.. Add the public IP ranges")
```

Begin with your dataset

Conditional One: Outgoing Traffic

Conditional Two: Incoming Traffic

Conditional Three: Internal Traffic

Default (1=1): Whatever Else

splunk> .conf2017

# Technique: Eval

## The searchmatch Function

- When it comes to applying business logic via eval, there are tons of options, mostly covered here: http://docs.splunk.com/Documentation/SplunkCloud/6.6.0/SearchReference/ConditionalFunctions
- A favorite of mine is searchmatch. I have seen it be slower than a highly optimized field-based approach, but it makes logic so easy that anyone can get started with it.
- What searchmatch will do is simply run a search, just as if you used the | search command, but within an eval if or case statement. Here are some examples:

```
sourcetype=what_have_you

| eval is_us = if(searchmatch("country: US"), 1, 0)

| eval do_errors_exist = if(searchmatch("error"), 1, 0)

| stats
      count
      sum(is_us)
      count(eval(searchmatch("type=important")
  by do_errors_exist
```

This *should* be extracted into a field, but if you haven't done it yet, you can use searchmatch.

Maybe we just need to know if a particular string is in the raw logs

We are just checking a field here, which you could do directly, but if you're not comfortable with more advanced methods yet, stay simple.
And of course we can embed this into stats - see eval + stats in this presentation for more here.

# Technique: Eval

## The replace Function

- Often we run into scenarios where you need to do string manipulation. In Splunk we often end up using | rex for these scenarios, as it can do regex field extraction and also sed search and replace. However, those are universal. With eval and replace, you can put this inside of a conditional.

```
sourcetype=what_have_you

| eval _raw=if(NOT searchmatch("country: US"),
        replace(_raw, "user=\S*", "user=XXXXX"),
        _raw)

| eval user=if(NOT searchmatch("country: US"),
        "XXXXX",
        _raw)
```

If this is not a US message, let's replace the username with a series of Xs in the raw log

Let's do the same thing with the username field.

We don't actually recommend enforcing field based anonymization this way due to tricky workarounds, but it is worth nothing how this is possible for some circumstances

# Technique: Eval

## The spath Function

- If you've ever had to deal with complicated JSON or XML, the eval spath function is a lifesaver. It is similar to the | spath command, but it can be embedded in conf files.

```
sourcetype=my_XML

| eval sender = spath(_raw, "envelope.header.sender")

| rex max_match=0 "(?<transaction>)<trans>.*?</trans>)
| mvexpand transaction

| eval payload=spath(transaction, "trans.body")
| eval payload_length = len(payload)

| table sender payload_length payload
```

We can extract XML or JSON values out of _raw logs

When we deal with very complicated json, mvfields become very important. We will cover that in Multi-Value fields, later in this presentation.

You can also apply this to individual fields, quickly and easily.

splunk> .conf2017

# Technique: Eval

## The where Search Command

- I know what you're saying - where is a search command, it's not eval. But a common question I get is how | search is different from | where. The big difference is that | where uses eval logic.
- Anything you would put into the conditional in an if statement, you can put into a where clause.

```
sourcetype=what_have_you

| where
    (
        country!="US"
        AND NOT searchmatch("country: US")
    )
   OR match(
        urldecode(query_string),
        "[rR]estricted")
```

You do have to use the more rigid eval type syntax here, but you can do some much more advanced logic.

Did you know you can do urldecoding (e.g., %23 -> #, %24 -> $, etc.)? And regex matching? All of that in a where clause.

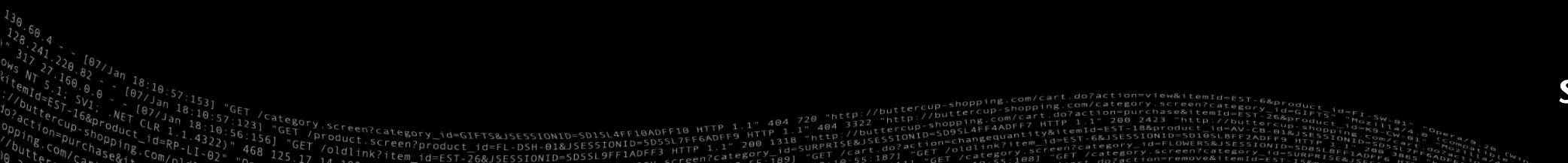
splunk> .conf2017

# Technique: Multi-Value Fields

## Background and Challenges

- "You have JSON, so your life is easy! Oh, did you say nested JSON? Eep…"
- "I want to tag events just for a specific search!"
- "I want to analyze IP occurrences, whether it's src_ip or dest_ip, I just care about ip"
- "For some reason I have a multi-value field… I want to analyze each field individually!"

- Multi-Value fields are a great swiss army knife inside of SPL, but they're also one of the least obvious techniques. Let's look at how they work.



# Technique: Multi-Value Fields

## JSON data

- Simple JSON data is very easy to deal with. Poorly structured data is a pain.

```
sourcetype=datasource1

| eval direction = case(
   cidrmatch("10.0.0.0/8", src_ip) AND NOT
            cidrmatch("10.0.0.0/8", dest_ip), "outgoing",
    NOT cidrmatch("10.0.0.0/8", src_ip) AND
            cidrmatch("10.0.0.0/8", dest_ip), "incoming",
    cidrmatch("10.0.0.0/8", src_ip) AND
            cidrmatch("10.0.0.0/8", dest_ip), "internal",
    1=1, "outgoing to outgoing.. Add the public IP ranges")
```

Begin with your dataset

Conditional One: Outgoing Traffic

Conditional Two: Incoming Traffic

Conditional Three: Internal Traffic

Default (1=1): Whatever Else

splunk> .conf2017

# Technique: Multi-Value Fields

## Tagging Events

- case is skipped by almost everyone who has never been a programmer. If you've been a programmer, you already know about it. If you haven't, get psyched.
- One of the greatest strengths of eval is that it allows you to embed all manner of business logic. Invariably, this requires you to have if statements… but often, you end up with multiple scenarios. If a, then x, if b, then y, if c, then z, if d, then throw an error.
- Many use nested if statements, but case handles multiple conditions with ease.

```
sourcetype=datasource1

| eval direction = case(
   cidrmatch("10.0.0.0/8", src_ip) AND NOT
            cidrmatch("10.0.0.0/8", dest_ip), "outgoing",
    NOT cidrmatch("10.0.0.0/8", src_ip) AND
            cidrmatch("10.0.0.0/8", dest_ip), "incoming",
    cidrmatch("10.0.0.0/8", src_ip) AND
            cidrmatch("10.0.0.0/8", dest_ip), "internal",
    1=1, "outgoing to outgoing.. Add the public IP ranges")
```

Begin with your dataset

Conditional One: Outgoing Traffic

Conditional Two: Incoming Traffic

Conditional Three: Internal Traffic

Default (1=1): Whatever Else

splunk> .conf2017

# Technique: Multi-Value Fields

## How Did I Get Here? How do I get out of Multi-Value land?

- Most commonly you have a multi-value field that you just want to split (e.g., two IP addresses that you want to split into two different events).
- This is easily done with mvexpand {field name}
- Keep in mind though that this will split *all* of the fields. If you only need a couple of fields, then use | fields beforehand to get rid of the others so that you don't consume excessive memory.
  - Splunk does try to deal with that stuff automatically, but I like to guide Splunk here.
- The other most common scenario I see is you have two values that are the same for a particular value. Usually this is a quirk of the data generator, but sometimes you will have the same value twice for every field.
- Two approaches for this scenario. The easiest (that I just learned!) is:
  | eval value=mvdedup(value)
- A slightly heavier but also more flexible approach is using streamstats, as you have all of the flexibility of stats:
  | streamstats window=1 values(value) as value values(eval(NOT match(value, "^\d")) as value2

splunk> .conf2017

# Technique: Multi-Value Fields

## When Two (or more fields) Become One

- One of the scenarios where I use multi-value fields decently often is to simplify source/dest analysis. In a typical perimeter NGFW log, you have src_ip, dest_ip, src_translated_ip, and dest_translated_ip. If I want to track the top # of IPs associated with IPS alerts, I can look for the top IPs without worrying as much about the directionality.

```
sourcetype=ngfw
| fields severity src_ip dest_ip src_translated_ip
dest_translated_ip

| eval ip = mvappend(src_ip, dest_ip,
src_translated, dest_translated_ip

| stats max(severity) count by ip
```

Here are the fiels that ultimately I care about

Conditional One: Outgoing Traffic

Conditional Two: Incoming Traffic

Conditional Three: Internal Traffic

Default (1=1): Whatever Else

splunk> .conf2017

# Technique: Multi-Value Fields

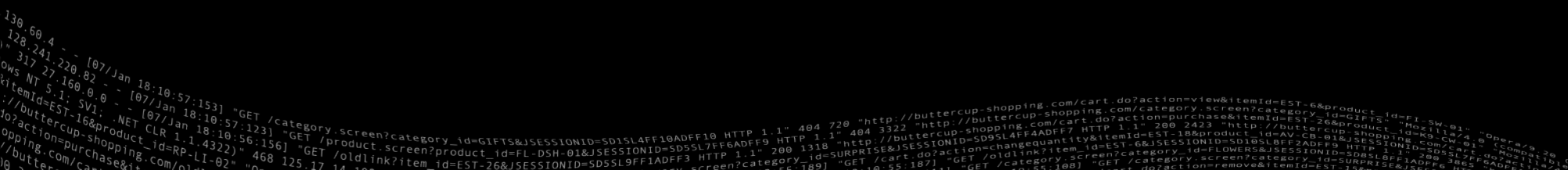
## Warning: Null Fields, the importance of coalesce

- This is general to working with eval, but I find it comes up often in the context of multi-value fields.
- Whenever there is the possibility that you might have a null value, make sure to coalesce it to something non-null, otherwise it could break everything.
- BAD: | eval description = "… Second Username (if present): " . mvindex(users, 1,1)
- GOOD: | eval description = ".. Second Username (if present): " . coalesce(mvindex(users, 1, 1), "N/A")

# Technique: Stats on Stats

## Background and Challenges

- Remember that as you build out a Splunk search each command sends results to the next, but all any search command takes as input is a series of fields. Even many intermediate searchers don’t take advantage of this capability!
- "I would like to track how many events occur per day per user, and then find anomalies in that daily trend."



# Technique: Stats on Stats

- We leverage the first stats to grab per day elements, and then the second stats to aggregate and analyze trends.

```
tag=authentication

| bucket _time span=1d
| stats  dc(dest) as count by user, _time

| stats count as num_data_samples
        max(eval(if(_time >= relative_time(now(),
            "-1d@d"), count,null))) as count
        avg(eval(if(_time<relative_time(now(),
            "-1d@d"), count,null))) as avg
        stdev(eval(if(_time<relative_time(now(),
            "-1d@d"), count,null))) as stdev
    by user
```

Start with whatever base search you want

The first stats will pull the unique number of destinations per user per day

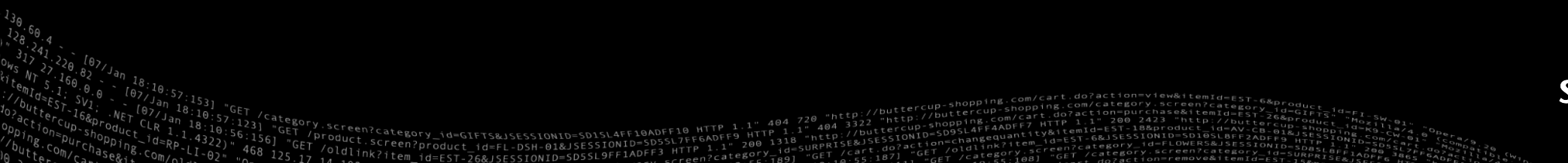
Now our second stats will calculate the last day's results, the average, and the stdev.

splunk> .conf2017

# Technique: Formatting a Table

## Background and Challenges

- My first smart phone was a Samsung BlackJack. I returned in 30 days in and bought a first gen 2G iPhone. They both had maps, they both had mobile web, but the iPhone didn't suck to use.
- You want results to stand on their own, and not be disregarded by a bad UI. For that, always format searches cleanly. This helps make searches actionable, and reduces the odds that important results will be overlooked.
- Avoid printing worthless information
- Always provide a drilldown capability.



splunk> .conf2017

# Technique: Formatting a Table

## Format within the SPL

- Use convert, eval, and table to clean up your output

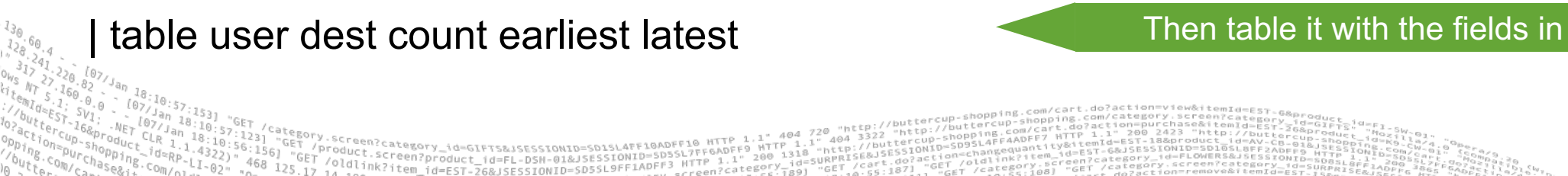
```
tag=authentication
| stats earliest(_time) as earliest
        latest(_time) as latest
        count
        by user, dest
| where earliest >= relative_time(now(), "-1d@d")

| convert ctime(earliest) ctime(latest)
          timeformat="%m/%d/%Y %H:%M:%S"

| eval dest=replace(dest, ".contoso.com", "")

| table user dest count earliest latest
```

Build whatever detection you are looking for, in this case looking for people logging to servers for the first time in the last day. For an example similar to this, check out the "Lookup Caching" technique, which scales really well.

Definitely don't print an epoch timestamp ever. But ever for normal timestamps, make sure that they match each other and what analysts are expecting. They have to get it *really* fast, so get in the habit

Maybe you have unnecessary info? Format it.

Then table it with the fields in a sensible order



splunk> .conf2017

# Technique: Formatting a Table

## Drilldown In the Worst Scenario

- Sometimes you have no clean drilldown capability, e.g., in an email alert. Even in that scenario, give a search string that can be run.

```
tag=authentication
| stats earliest(_time) as earliest latest(_time) as
latest count values(sourcetype) as sourcetypes
values(indexes) as indexes by user, dest
| where earliest >= relative_time(now(), "-1d@d")
| eval drilldown= "index=" . mvjoin(indexes, " OR
index=") . " sourcetype=" . mvjoin(sourcetypes, "
OR sourcetype=") . " user=" . user . " dest=" .
dest . " earliest=" . earliest . " latest=" . latest
| convert ctime(earliest) ctime(latest)
            timeformat="%m/%d/%Y %H:%M:%S"
| eval dest=replace(dest, ".contoso.com", "")
| table user dest count earliest latest drilldown
```

Most of this search was already covered - I've grayed out those parts for clarity.

We've now added sourcetypes and indexes into our base search.

There's not a ton of complexity here - we're just composing a big string that someone could copy-paste. Not the mvjoin to handle many different potential sourcetypes or indexes, though.

In your final table, you can include the drilldown but exclude the ugly other fields that it is composed of. This lets analysts just copy-paste, as an item of last resort.

splunk> .conf2017

# Technique: Formatting a Table

- Building a dashboard? Print something good, and then drilldown well.

```
<panel>
  <title>Users logging into new servers (with drilldown)</title>
  <table>
    <search base="the last slide to save space, but add index
and sourcetype">
      <query> | table sourcetype index user dest count
earliest latest drilldown | sort - count </query>
    </search>
    <fields>["user", "dest", "count", "earliest","latest"]</fields>
    <drilldown>
      <link>/app/search/search?q=index=……..</link>
    </drilldown>
  </table>
</panel>
```

Here we have a search with a few fields that we want to use for drilldown, but don't want to actually show to the analyst.

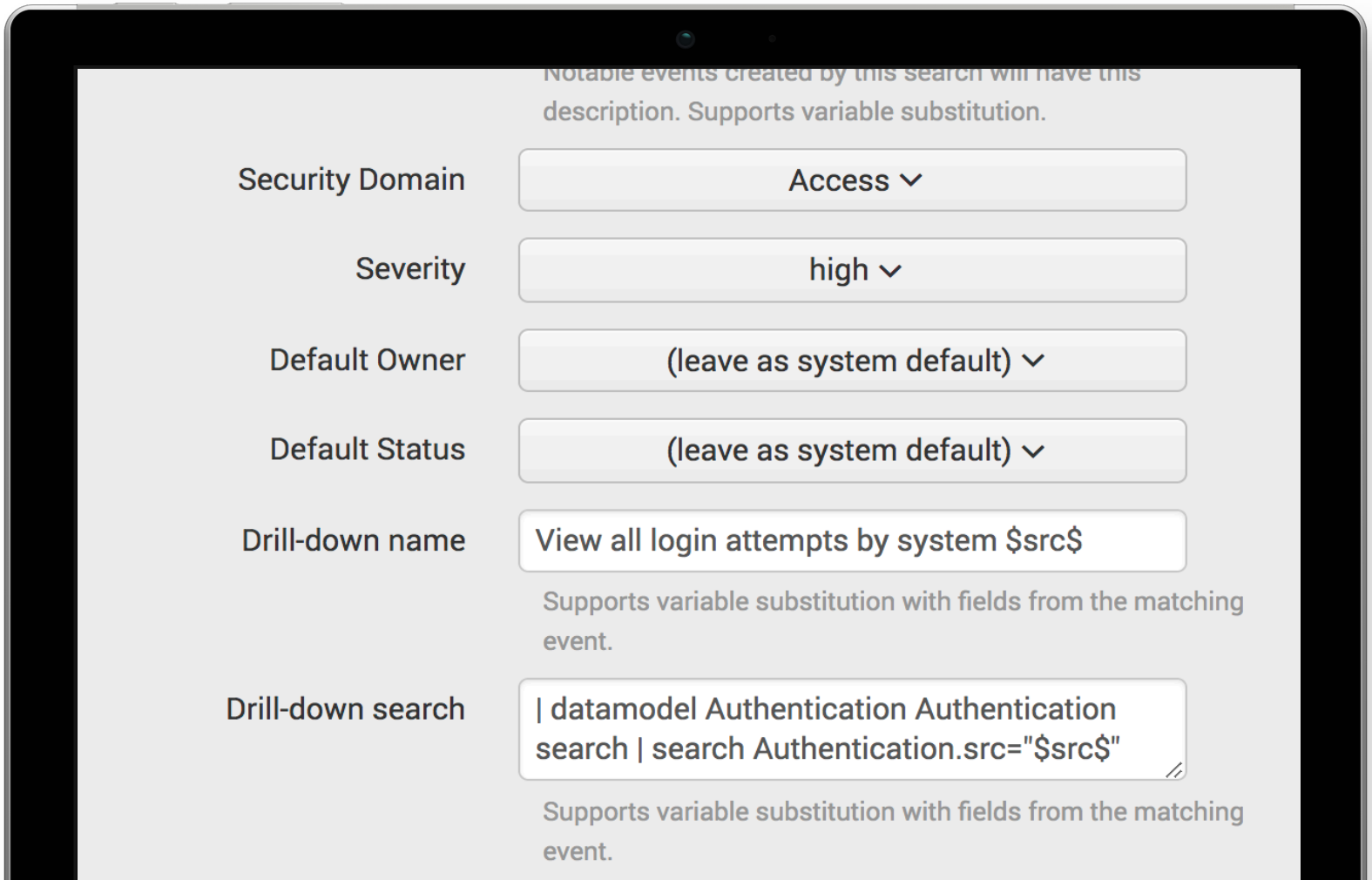
<fields> controls what is shown. (json format..) Specifically, we are not showing the drilldown field

Now we can use the <drilldown> <link> to define the actual search. This can be weird if you're not familiar with URL Encoding - it's easiest to just google it. But here we are opening in the search app, search view, and passing the query (q=). Then we URL Encode the actual fields we want to put in there.

splunk> .conf2017

# Technique: Formatting a Table

## Or just use our out of the box tools..




- … or just use ES with it's built-in tables and built-in drilldown searches… which is way way easier

splunk> .conf2017

# Technique: Formatting a Table

## Working Example

- From conf2016 Security Ninjutsu Part Three, a large customer shared a search that looks for scenarios where svchost.exe wasn't owned by services.exe. Cool search, yeah? Those MD5s are known legit svchost.exe versions in their environment. But this is going to the SOC, so what did they end with? A table.

```
sourcetype=Win*Security EventID=4688 BaseFileName="svchost.exe" NOT CreatorProcessName="services"

NOT (MD5="54A47F6B5E09A77E61649109C6A08866" OR […])

| sort 0 -_time

| table _time, Computer, SubjectDomainName, SubjectUserName, BaseFileName, CommandLine, CompanyName, CreatorProcessName, NewProcessName, FileDescription, FileVersion, MD5
```

splunk> .conf2017

# Technique: Multi-Scenario Alerts

## Background and Challenges

- "There are six different reasons why I might want this to alert, but I don't want to have six different searches!"
- "I have too many searches looking at the exact same dataset!"

- When we help people consolidate alerts from legacy SIEMs, there are generally two different types of consolidations. One is data sources, where our CIM allows us to avoid duplicating rules for different data types. The other is that we can consolidate rules.
- As an example, I looked at one SIEM dataset that had 54 different auth rules, where there were 7 different data sources and 8 different logic pieces. Those fit easily into just two Splunk searches.

# Technique: Multi-Scenario Alerts

## Actual SPL

- If you have something you want to tell analysts, tell them. You can put it in the playbook if you know they will always look at the playbook.. Otherwise embed it.

```
index=risk earliest=-30d

| stats values(source) as search_names
        sum(risk_score) as thirty_day_risk
        sum(eval(if(_time > relative_time(now(),
           "-1d"),risk_score,0))) as one_day_risk
    by risk_object

| eval threshold_1day = 500, threshold_30day = 1200
| eventstats avg(thirty_day_risk) as avg_thirty_day_risk
stdev(thirty_day_risk) as stdev_thirty_day_risk

| where one_day_risk>threshold_1day OR
         thirty_day_risk>threshold_30day OR
         thirty_day_risk>
             (avg_thirty_day_risk + 3 * stdev_thirty_day_risk)
```

This examples uses the ES Risk Framework

Using stats + eval, we can pull out many different metrics here. Slicing and dicing by data type or particular field value, all very easy.

Here we are using a mix of static thresholds, and behavioral thresholds calculated via eventstats. Eventstats is also helpful for augmenting analysis, just make sure not to exceed its memory limits, as it will silently fail.

Finally we can trigger on multiple different conditions with ease.

If you use multi-scenario alerts, make sure you have inline comments that explain the logic. See the next section!

# Technique: Multi-Scenario Alerts

## How complex is too complex?

- A word of warning here: I love multi-scenario alerts, because I am an SPL nerd. Most seasoned PS folks will probably tell you to avoid them, because often each alert ties to a different playbook an analyst would have to pursue. Or worse, an analyst would look at the alert and not really know what it means (emphasis: inline comments is the next section). Or even, it can allow you to create hundreds of effective rules, which we know often leads to bad security practices.
- There's fairly broad agreement on the risk example, because it is functionally doing something pretty straightforward (looking at risk indicators) and just tries to account for quick bursts, but also slow and low activity.
- Just be wary when creating these that you don't allow your newfound power to create an unhappy SOC.

# Technique: Inline Comments

## Background and Challenges

- It’s very easy to build advanced logic in correlation searches that are difficult for an analyst to quickly ascertain the meaning of. This results in comments like "I don’t know what to do with this" or "this is not actionable."
- Scenario One:
  - It’s very easy in Splunk to combine many different searches into one, but then analysts don’t know why it’s actually alerting.
  - For example, in analyzing the risk framework, we can alert on slow and low, or short term burst activity, or do behavioral detections all in one search. But you need to tell the analyst what to look at.
- Scenario Two:
  - There can be some information that you would expect to be there, but maybe it’s just not. Tell the analyst so they don’t boggle.

splunk> .conf2017

# Technique: In Line Comments

## Simple Comments

- If you have something you want to tell analysts, tell them. You can put it in the playbook if you know they will always look at the playbook.. Otherwise embed it.

```
[... base search here …]

| eval "Remote Source Address"="It would sure be nice if the F5 told us where connections were coming from"

| rename dest_ip as "Local Destination Address"
         user as User

| table _time "* Address" Sourcetype
```

Start with whatever base search you want

Clue Analysts into what's going on here, so they know what to look for, if you cannot provide it. *Note* -- the ES Adaptive Response can help here, by adding related search results to your ticket.

BTW - rename your fields so that they make sense to the analysts. Try to be consistent across your searches, but don't make people divine what you mean by "outgoing_ip"

Yeah, of course we finish with a table

# Technique: In Line Comments

## Advanced Logic begets Advanced Comments

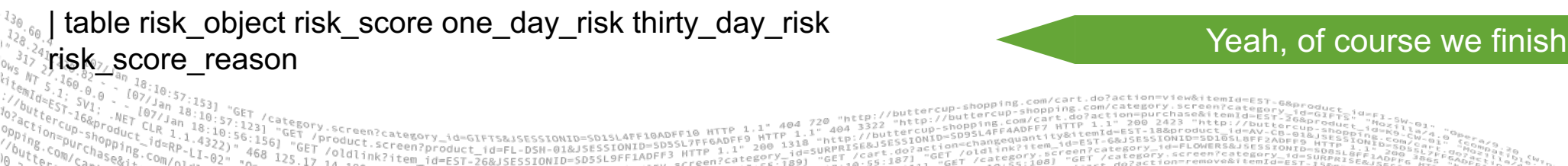
```
index=risk earliest=-30d | stats values(source) as search_names
sum(risk_score) as thirty_day_risk sum(eval(if(_time >
relative_time(now(), "-1d"),risk_score,0))) as one_day_risk by
risk_object | eval threshold_1day = 500, threshold_30day = 1200 |
eventstats avg(thirty_day_risk) as avg_thirty_day_risk
stdev(thirty_day_risk) as stdev_thirty_day_risk

| where one_day_risk>threshold_1day OR
thirty_day_risk>threshold_30day OR
thirty_day_risk>(avg_thirty_day_risk + 3 * stdev_thirty_day_risk)

| eval risk_score_reason = case(one_day_risk>threshold_1day, "One
Day Risk Score above " . threshold_1day,
thirty_day_risk>threshold_30day . " on " . strftime(now(), "%m-%d-
%Y"), "Thirty Day Risk Score above " . threshold_30day, 1=1, "Thirty
Day Risk Score more than three standard deviations above normal
(>" . round((avg_thirty_day_risk + 3 * stdev_thirty_day_risk),2) . ")") |
fields - avg* stdev*

| table risk_object risk_score one_day_risk thirty_day_risk
risk_score_reason
```

- If you're going to put in advanced logic, make sure you have advanced comments and explanations

We have three potential reasons why this alert would fire - one day risk, 30 day risk, or a behavioral risk. (Note: I think this behavioral risk is pretty weak..)

We had three conditions in the where, so we have 3 conditions to cover in the comment. Note that the conditionals are the same in the case statements. Fun fact: when combining searches with this method, the comment block will usually be way way longer..

Yeah, of course we finish with a table



splunk> .conf2017

# Technique: In Line Comments

Wait, *you* are telling me to keep my comments short? Have you looked at the deck you're writing right now? I have a mirror for you!

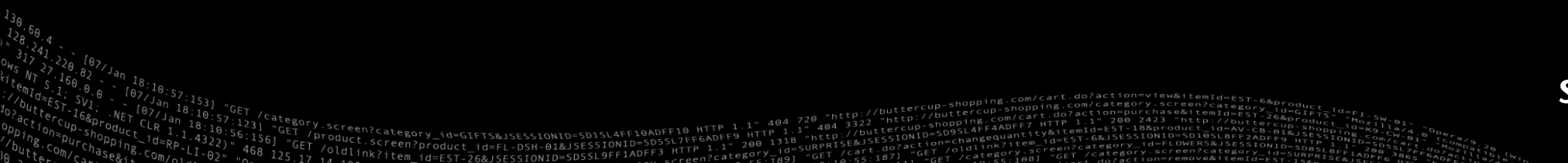
- You're familiar with tl;dr, right? Hard learned lessons:
- If your comment is more than a few words, people aren't going to read it.
- If you have 3 different potential comments, make them visually very different (e.g., don't start each with "This alerts because") so that people will notice the unfamiliar format at a glance.

# Technique: Tuning

## Background and Challenges

- "I like this correlation search but it generates too much noise"
- "I like that you tuned this correlation search, but wow is it ever long!"
- "I like that you tuned this correlation search, but you missed XYZ"

- Tuning searches is inevitable for correlation searches. Let's look at techniques for doing this scalably.



# Technique: Tuning

## Just toss that inline!

- By far, the easiest way to do tuning is to add the items inline. But beware, you can end up with super, super long searches.

```
sourcetype=Win*Security EventID=4688
BaseFileName="svchost.exe" NOT
CreatorProcessName="services"

NOT (MD5="54A47F6B5E09A77E61649109C6A08866"
OR […])

| sor

| table _time, Computer, SubjectDomainName,
SubjectUserName, BaseFileName, CommandLine,
CompanyName, CreatorProcessName,
NewProcessName, FileDescription, FileVersion, MD5
```

That's Tuning!

```
index=bro_smtp (("Workspace Webmail") OR ("Roundcube Webmail")) OR (  (wire subject=*wire* ((transfer subject=*transfer*) OR (funds subject=*funds*)))  OR (remediate subject=*remediate*)  OR (remit* subject=*remit*) ) | rex field=to
mode=sed "s/(\"[^,]*),([^,]*\")/\1_\2/g"| makemv delim="," to| mvexpand to| rex field=mailfrom "\<(?<mailfrom>[^\>]+)\>"| rex field=sender "\<(?<sender>[^\>]+)\>"| rex field=recipient "\<(?<recipient>[^\>]+)\>"| rename sender as from,
recipient as rcptto| rex field=to "\<(?<to>[^\>]+)\>"| regex to="@" | rex field=reply_to "\<(?<reply_to>[^\>]+)\>"| rex field=mailfrom "@(?<mailfrom_domain>.*)$"| rex field=mailfrom_domain "(?<mailfrom_basedomain>[^\.]+\.[^\.]+)$"| rex
field=from "@(?<from_domain>.*)$" | rex field=from_domain "(?<from_basedomain>[^\.]+\.[^\.]+)$"| rex field=reply_to "@(?<reply_to_domain>.*)$"| rex field=reply_to_domain "(?<reply_to_basedomain>[^\.]+\.[^\.]+)$"| eval
from=lower(from)| eval to=lower(to)| eval rcptto=lower(rcptto)| eval reply_to=lower(reply_to)| eval mailfrom=lower(mailfrom)
•	Munge your data and do any field extractions

| eval exclude=if(match(subject,"Wire Transfer Approval Required [A-Z]{2}\d{6}"),1,0)| eval exclude=if(match(subject,"WIRE TRANSFER Notification:.*, File \#\d+, Expense Report \#\d+"),1,0)| eval exclude=if(match(subject,"Infoworks Notification
\d{9}"),1,0)| eval exclude=if(match(subject,"(Gg|roupon"),1,0)| eval exclude=if(match(mailfrom,"(Gg|roupon"),1,0)| eval exclude=if(match(from,"(Gg|roupon"),1,0)| eval exclude=if(match(subject,"Delivery Status Notification (Failure)"),1,0)| search
(to=*[domain]* OR to=*[domain]*)AND NOT( mailfrom=*@bounce.secureserver.net OR mailfrom="dt.pixture@gmail.com"  OR mailfrom=*@googlegroups.com  OR mailfrom=postmaster@urs.com  OR mailfrom=*@[domain].com  OR
mailfrom=*@mail.cignasecure.com  OR mailfrom=*@[domain].com OR from=*@[domain].com  OR from=*@[domain].com OR from="*.groupon.com" OR from="service@remitly.com" OR from=*schneider-electric.com OR mailfrom=*@[domain]
OR mailfrom=*@brindyl.com  OR mailfrom=*.[domain].com OR mailfrom=*@mvfax.com OR mailfrom=*@observium.org OR mailfrom=*@xansys.org OR from=*@app.topica.com  OR from=info@securemailingtodav.com  OR to=*@[domain].com
OR to=agencies@[domain].com OR to=observium@observium.org  OR (reply_to="-" AND mailfrom="<>" AND (from=*[domain].com OR from=*[domain].com)) OR (exclude=1 AND (to=*[domain].com OR to=*[domain].com) AND
(from=*[domain].com OR from=*[domain].com))) AND NOT ( "you're pre-approved"  OR "chance to win"  OR "your preapproved"  OR "your pre-approved"  OR "splunk alert" OR "Delivery Status Notification (Failure)")
•	Exclude whitelisted / tuned sources

| eval phish_score=1| eval phish_score=case(reply_to==from,phish_score,reply_to=="-",phish_score,1==1,(phish_score*10)+1) | eval phish_score=case(reply_to_domain==from_domain,phish_score,reply_to==""-
",phish_score,1==1,(phish_score*10)+1) | eval phish_score=case(reply_to_basedomain==from_basedomain,phish_score,reply_to=="-",phish_score,1==1,(phish_score*10)+1) | eval
phish_score=if(mailfrom_domain==from_domain,phish_score,(phish_score*10)+1) | eval phish_score=if(mailfrom_basedomain==from_basedomain,phish_score,(phish_score*10)+1) | eval
phish_score=if(mailfrom==from,phish_score,(phish_score*10)+1)| eval phish_score=if(match(mailfrom_domain,"(closer namealike pattern1)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"(closer namealike
pattern1)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"(looser namealike pattern3)"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"(closer namealike
pattern4)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"(closer namealike pattern5)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"(looser namealike
pattern6)"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"(closer namealike pattern7)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"(closer namealike
pattern8)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(mailfrom_domain,"(looser namealike pattern9)"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(from_domain,"(closer namealike
pattern1)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"(closer namealike pattern2)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"(looser namealike
pattern3)"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(from_domain,"(closer namealike pattern4)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"(closer namealike
pattern5)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"(looser namealike pattern6)"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(from_domain,"(closer namealike
pattern7)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"(closer namealike pattern8)"),(phish_score*100)+1,phish_score)| eval phish_score=if(match(from_domain,"(looser namealike
pattern9)"),(phish_score*10)+1,phish_score)| eval phish_score=if(match(from_domain,"(yahoo|gmail|outlook|hotmail|freemail|hushmail|googlemail|live\.com|msn\.com)"),(phish_score*10)+1,phish_score)| eval
phish_score=if(match(reply_to_domain,"(yahoo|gmail|outlook|hotmail|freemail|hushmail|googlemail|live\.com|msn\.com)"),(phish_score*10)+1,phish_score)| eventstats count(subject) as spammy by subject| eval
targeted_score=phish_score*(1/spammy)
•	Build threshold

| where targeted_score>1000000 AND phish_score>10000000 AND spammy<24| table _time mailfrom from replyto rcptto to subject user_agent spammy phish_score targeted_score| sort -targeted_score, subject, -phish_score, mailfrom, _time
•	Filter out malicious content

•	This search, printed out in its formatted form, is 101 lines long. While the logic is highly developed and tuned, it's functionally no different than any other search.
```

That's *also* Tuning!

splunk> .conf2017

# Technique: Tuning

## Make a Macro

- One place to look, can be applied to multiple different searches, and makes for short non-scary searches!

In macro config (macros.conf, or Web UI):
[standard_host_exclusions]
NOT (host=vuln-scan*.mycompany.local OR
host=*.old-env.mycompany.local OR {…} )

Define your exclusions in macros.conf or in Settings -> Advanced Search -> Macros

In Correlation Search:
index=authentication `standard_host_exclusions`
| rest of your correlation search

Add your macro into the correlation search

# Technique: Tuning

## Build a Lookup Table that the SOC can access

- Define a lookup table with the fields you care about, then bring it into the search. The SOC can then access that lookup table and update it.

Create a lookup field with whatever field you care about, e.g., standard_host_exclusions.csv:

```
host
vuln-scan*.mycompany.local
*.old-env.mycompany.local
```

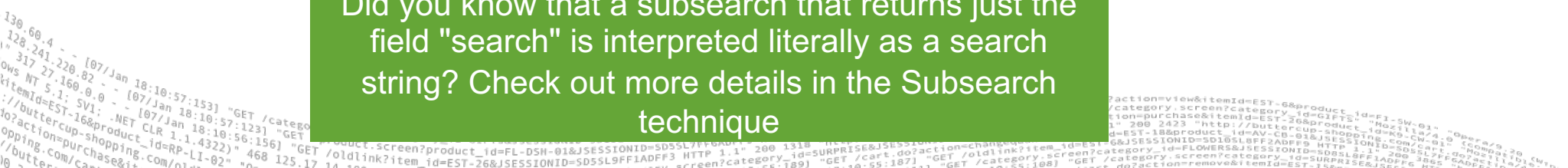
This is just a simple CSV. You can even allow the SOC to update this via the lookup editor app (or the built in capability in Enterprise Security)!

Then bring that into your correlation search:

```
tag=authentication
    [.  | inputlookup standard_host_exclusions.csv
        | stats values(host) as search
        | eval search="NOT (host=" . mvjoin(host, " OR host=") . ")"
    ]
```

This allows you to take those hosts, and then craft a NOT (…) string like in the macro example that is generated the moment that you click "search" with almost no performance impact.
P.S. You can put that in a Macro too!

Did you know that a subsearch that returns just the field "search" is interpreted literally as a search string? Check out more details in the Subsearch technique



splunk> .conf2017

# Technique: stats + eval

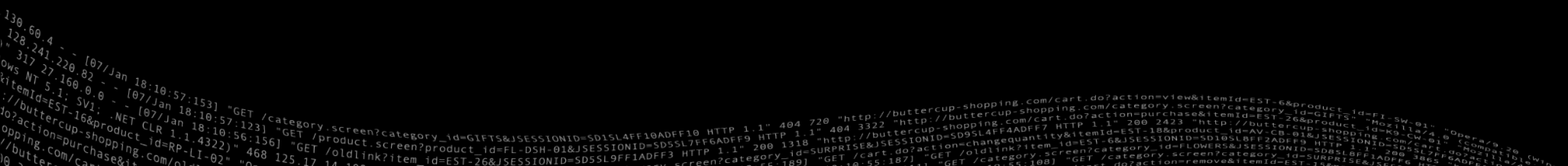
## Background and Challenges

- If you remember only one thing in this entire presentation, remember this.

Stats + Eval is the most powerful tool for Splunk Correlation Searches

Stats + Eval = BEAST MODE

- "I want to alert if we see a set of events across 7 different sourcetypes in a particular order"
- "I want to alert if we see a particular error coming from the web server right after a user submits a particular request."
- "I want to run a search that tracks multiple different thresholds based on different time windows to find slow and low activity while also finding bursts oh and I also want to implement a specific piece of custom logic."



# Technique: stats + eval

- You can input a lookup, then output a lookup, and then continue on your search. Run this search every day/hour, and take advantage of a 90 day baseline!

```
tag=authentication
 | stats count(eval(action="success")) as successes
         count(eval(action="failure")) as failures

         values(eval(if(action="success",user,null))) as
             "Successful Users"

         count(eval(if(searchmatch("example of log
             message"), 1, null))) as "example hits"

         count(eval(if(match(email,
             "\@buttercupgames\.com"),1,null))) as
             buttercup_emails
     by user
```

Start with whatever base search you want

Using stats + eval allows you to create columns based on individual items.

You can even use if statements, and use null for items you don't want to include.

Searchmatch even allows you to execute a raw search within the eval!

I'm also fond of regex matching inside of stats, 'cause you can just do that!

splunk> .conf2017

# Technique: stats + eval

## Example from .conf 2015

- Joins are really computationally expensive, and limited
- Only if you have one *very* rare term search and one dense search, are subsearches a great approach. (Best if they're not IP based, because IP searches are challenging)
- **Incorrect (10k results!)**: tag=malware action=allow | stats count as infections by host | join host [search index=proxy category=uncategorized | stats count as hits by host]
- Maybe Incorrect (**400 seconds, 10k malware hits**): [search tag=malware action=allowed | dedup dest | rename dest as src | table src]  (tag=proxy category=uncategorized)  | stats count(eval(tag="malware")) as NumMalwareHits count(eval(tag="proxy")) as NumProxyHits by src
- Better (**72 seconds**): (tag=malware action=allowed) OR (tag=proxy category=uncategorized) | eval mydest=if(tag="malware", dest, src) | stats count(eval(tag="malware")) as malware count(eval(tag="proxy")) as proxy by mydest | where malware>0 AND proxy>0
- Best (**14 seconds**): | tstats prestats=t summariesonly=t count(Malware_Attacks.src) as malwarehits from datamodel=Malware where Malware_Attacks.action=allowed groupby Malware_Attacks.src | tstats prestats=t append=t summariesonly=t count(web.src) as webhits from datamodel=Web where web.http_user_agent="shockwave flash" groupby web.src | rename web.src as src Malware_Attacks.src as src | stats count(Malware_Attacks.src) as malwarehits count(web.src) as webhits by src| where malwarehits > 0 AND webhits > 0

```
count(eval(tag="malware")) as malware count(eval(tag="proxy")) as proxy by dest
```

tstats is awesome! Check out the tstats section of this presentation

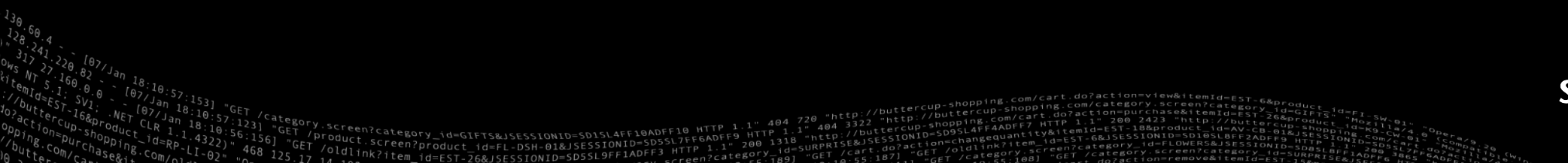
splunk> .conf2017

# Technique: Override Urgency/Severity/Risk

## Background and Challenges

- "I think this is generally low severity, unless it happens for John Smith or his team of Research Scientists, in which case OH NO!"
- "We can usually ignore this assuming it's not happening to our VIPs"

- This comes up most frequently for ES customers, but can be applied to anyone else, depending on how you handle your upstream ticketing.
- Inside of ES, we have default severity, and default risk indicators. In addition, Urgency is automatically calculated based on the combination of severity and the priority of the asset or identity involved. But all of those can be overridden from within the search, to let you prioritize (or de-prioritize) any particular events or users.



# Technique: Override Urgency/Severity/Risk

- If you include the fields urgency, severity, risk_object, risk_object_type, or risk_score, it will override whatever default values exist.

```
tag=authentication
| filter_for_bad_stuff

| lookup user org_risk OUTPUT NumRisk

| eventstats avg(NumRisk) as AvgNumRisk


| eval risk_score=round(40*(NumRisk / AvgNumRisk), 0)
| eval risk_object=if(user="administrator", "John Smith")
| eval risk_object_type="user"
| eval severity=if(risk_score>120, "critical", "medium")
```

Run your detection logic.

Use lookup, or anything else necessary to add context to the event.
In this case, the lookup gives us a numeric risk coefficient (probably 1-5). Because we want to hardcode very little, we calculate the avg coefficient.

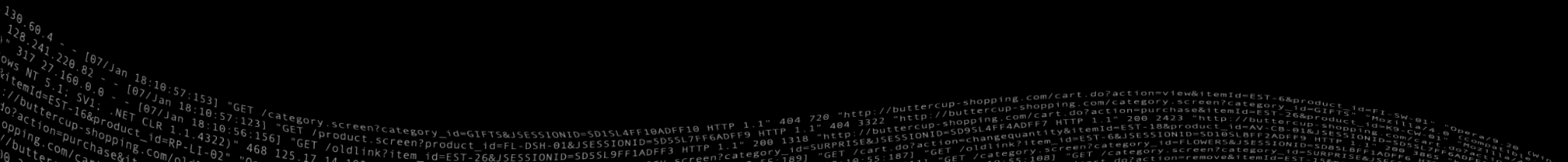
Finally, we can hardcode the risk_score, risk_object, risk_object_type, and severity. You can technically hardcode the urgency as well, though usually that shouldn't be necessary unless you haven't configured your assets correctly.

splunk> .conf2017

# Technique: Common Apps

## Background and Challenges

- "I want to conquer the world with Splunk Apps"



splunk> .conf2017

# Technique: Common Apps

## Splunk Security Essentials

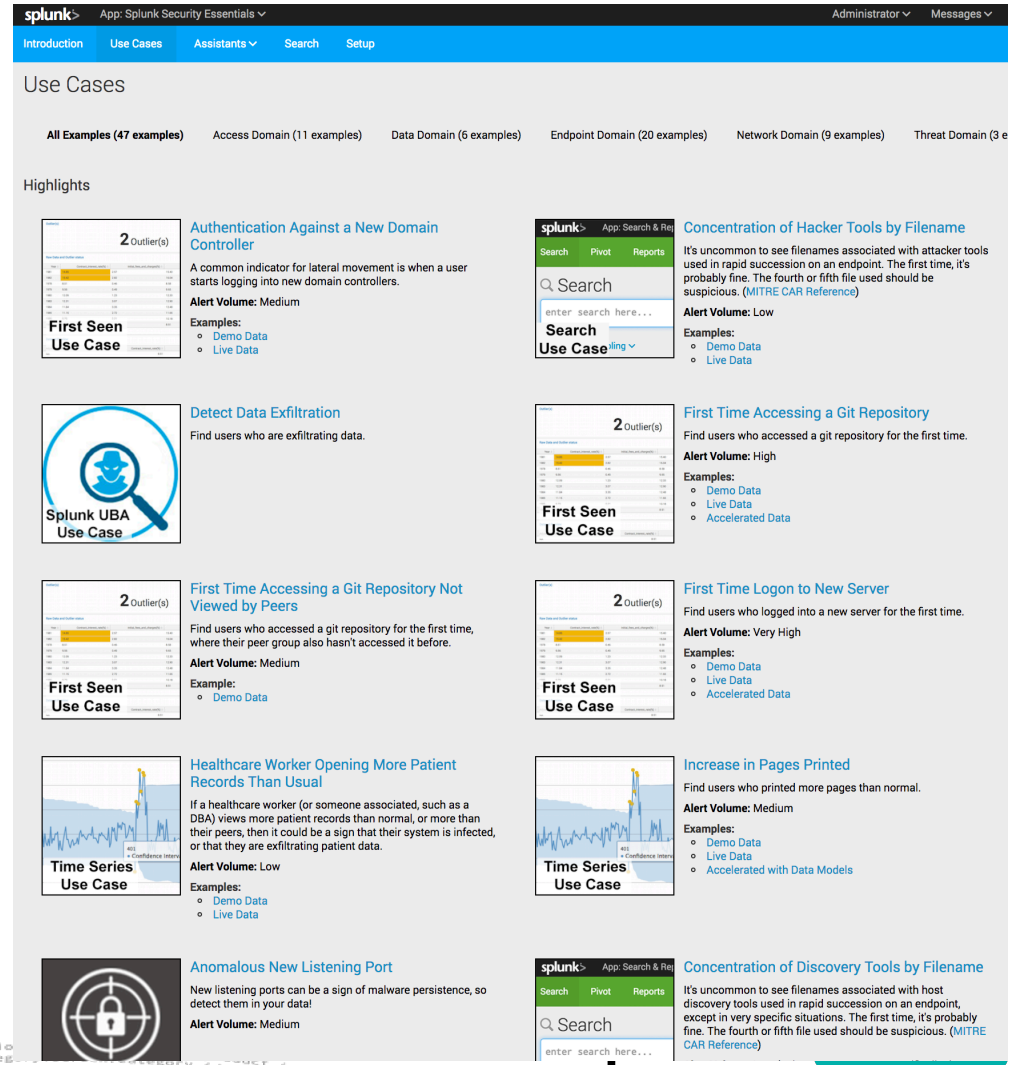
https://splunkbase.splunk.com/app/3435/

Identify bad guys in your environment:

- ✓ 50+ use cases common in UEBA products, all free on Splunk Enterprise
- ✓ Target external attackers and insider threat
- ✓ Scales from small to massive companies
- ✓ Save from the app, send results to ES/UBA

The most widely deployed UEBA product in the market is Splunk Enterprise, but no one knows it.

**Solve use cases you can today for free, then use Splunk UBA for advanced ML detection.**



# Technique: Common Apps

## URL Toolbox

https://splunkbase.splunk.com/app/2734/

- DNS exfil detection - tricks of the trade
- parse URLs & complicated TLDs (Top Level Domain)
- calculate Shannon Entropy
- List of provided lookups
  - ut_parse_simple(url)
  - ut_parse(url, list) or ut_parse_extended(url, list)
  - ut_shannon(word)
  - ut_countset(word, set)
  - ut_suites(word, sets)
  - ut_meaning(word)
  - ut_bayesian(word)
  - ut_levenshtein(word1, word2)

```
index=bro sourcetype=bro_dns query=nsa.gov.openwifi.defcon.org| head 1
| `ut_parse(query)`| `ut_shannon(ut_subdomain)` |fields url ut*|transpose
```

✓ 1 event (before 3/9/16 4:03:16.000 PM)

Events | Patterns | Statistics (24) | Visualization

20 Per Page ▾ | Format ▾ | Preview ▾

| column | row 1 |
|---|---|
| url | nsa.gov.openwifi.defcon.org |
| ut_domain | defcon.org |
| ut_domain_without_tld | defcon |
| ut_fragment | None |
| ut_netloc | nsa.gov.openwifi.defcon.org |
| ut_params | None |
| ut_path | None |
| ut_port | 80 |
| ut_query | None |
| ut_scheme | None |
| ut_shannon | 3.5 |
| ut_subdomain | nsa.gov.openwifi |
| ut_subdomain_count | 3 |
| ut_subdomain_level_1 | openwifi |
| ut_subdomain_level_2 | gov |
| ut_subdomain_level_3 | nsa |
| ut_tld | org |

# Technique: Common Apps

## Common URL Toolbox Usages

- Checking Randomness via Entropy. Random characters in filenames or domain names can indicate suspicious behavior! It can also create false positives (CDNs, etc.)
  - sourcetype=win*security EventCode=4688 Users New_Process_Name=*\Users* | stats count by New_Process_Name,host | lookup ut_shannon_lookup word as New_Process_Name | rename ut_shannon as "Shannon Entropy Score" New_Process_Name as Process,host as Endpoint
- Checking for similar strings can be useful particularly to find email phishing. Levenshtein gives us the distance between two strings.
  - sourcetype=proxy | stats count by domain | eval list="mozilla", mydomain="mycompany.com" | `ut_parse_extended(domain, list)` | lookup ut_levenshtein_lookup word1 as ut_domain word2 as mydomain | where ut_levenshtein < 3
  - Look for the Levenshtein-Damerau algorithm in JellyFisher (next slide) that better supports out-of-order characters

splunk> .conf2017

# Technique: Common Apps

## URL Toolbox Split into URL Parser and Jellyfisher

### URLParser

- Took the url parsing capabilities in URL Toolbox and rewrote them so that they are lightning fast.
- Lacks any of the statistical capabilities, but parsers very fast
- Separate from "URL Parser" (with a space) which is older and doesn't use new Splunk capabilities
- https://splunkbase.splunk.com/app/3396

### JellyFisher

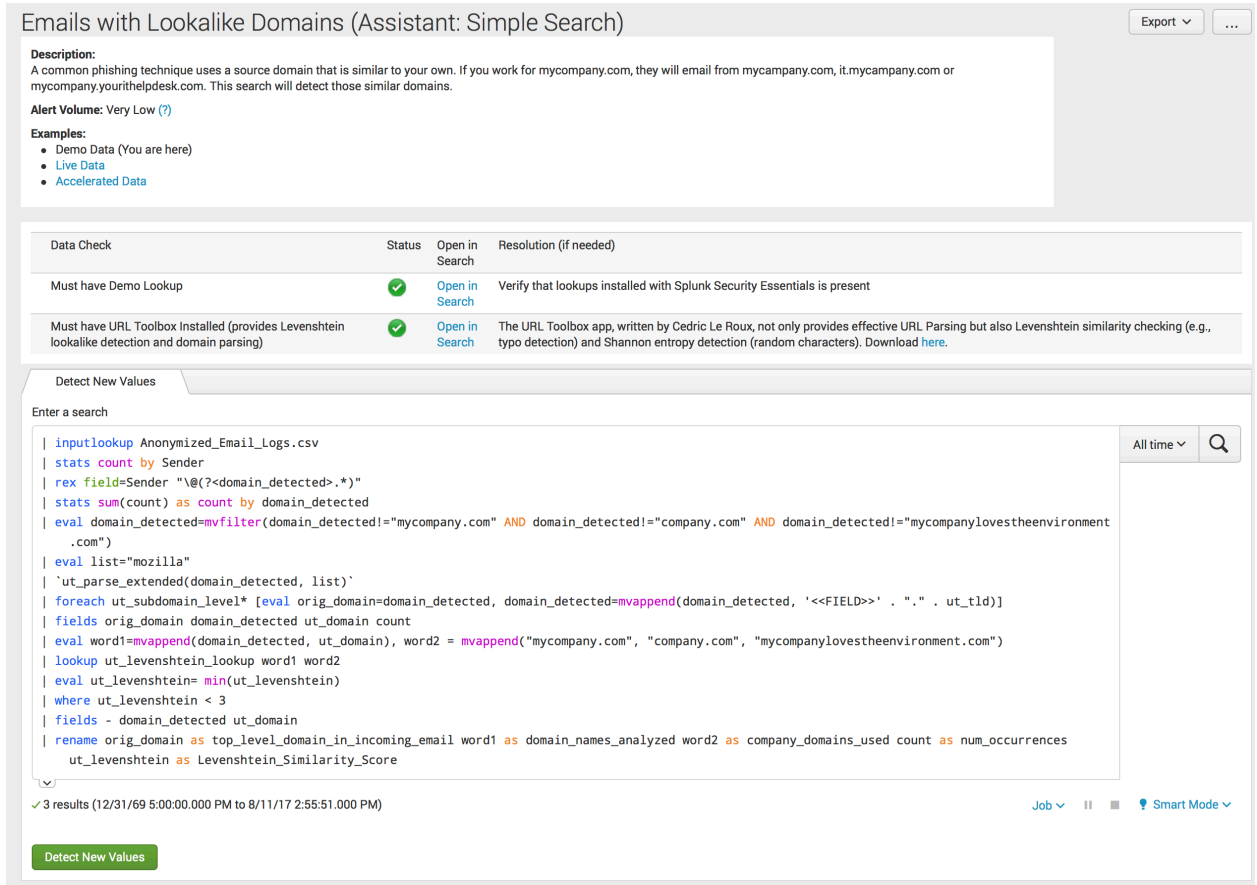
- Took the statistical capabilities from URL Toolbox and built them brand new with a super fast library
- Lots of new capabilities, such as phonetic matching (kirt vs curt)
- Brand new as of Aug 2017, uses the JellyFish library
- https://splunkbase.splunk.com/app/3626

# Technique: Common Apps

## Working Example

- Splunk Security Essentials demonstrates both Entropy and Levenshtein via URL Toolbox

1. Download the app off Splunkbase
2. Open: Emails with Lookalike Domains
3. This use case supports multiple different internal domains
4. Enjoy that you do not have to write this SPL!




splunk> .conf2017

# Technique: Risk

## Background and Challenges

- "When I look at my alerts, I know which people I really need to care about first - now why can't my Splunk alerts show the same thing?"
- Splunk Enterprise Security has a couple of specific risk mechanisms out of the box:
  - Asset / Identity Priority - this is closest to the need described above. If a medium severity rule fires on a PCI Database Server and the front desk Kiosk at the same time, one will show as low risk, the other as critical if you've defined one as a critical priority asset and the other as low. The same thing works for users.
  - Risk Framework - this allows you to assign a numeric risk score to each correlation alert, and then track the amount of risk incurred by each user, system, virus signature, etc. etc. You can even create notable events from this data!
- Some customers don't have ES (yet!) and but still need to prioritize events.

Disclaimer: This use case was covered in .conf2015 Security Ninjutsu Part Two

# Technique: Risk

## Requirements for a Numeric Risk Register

- You hopefully know the high risk, high exposure users in your organization.
  - Sys Admins, Executives, Contractors
  - First 3 months of employment, last 3 months of employment
  - Have accessed a particular file share
- Sources:
  - AD Group Membership
  - AD Title
  - HRIS Employment Status
  - Audit Logs
- Implementation. Run a periodic search that:
  - Refreshes AD (or consolidates multiple ADs, etc.)
  - Initializes risk=1 for all users
  - Does a ton of evals to apply your logic, adding to risk
  - Outputs to a new lookup

# Technique: Risk

## Build Your Risk Lookup

- Apply our business logic to figure out how risky each person is in our org.

```
| inputlookup LDAPSearch
| eval risk = 1

| eval risk = case(NumWhoReportIn>100,
risk+10, risk)
| eval risk = case(like(Groups,
"%OU=Groups,OU=IT Security,%"), risk + 10,
risk)
| eval risk = case(like(title, "VP %"), risk+10,
like(title, "Chief %"), risk+100, 1=1, risk)

| fields risk sAMAccountName
| outputlookup RiskPerUser
```

Start by initializing Risk for all your users

Then apply your business logic to figure out what risk potential should be applied to each person.

To consider ways to define risk, think of questions like "how would I feel if someone from a particular department had a dispute and left the company" and then "why?"

Larger organizations may have a more mature process here

Finally, put this risk score into a lookup

# Technique: Risk

## Use Your New Risk Lookup

- Now that we have a risk lookup, we can apply it to any search

```
[… insert your Correlation Search …]

| stats count by user

| lookup RiskPerUser sAMAccountName as user

| eval AggRisk = risk * count

| eval DescriptiveRisk = case(AggRisk > 100,
"very high", AggRisk>30, "medium", AggRisk>5,
"low", 1=1, "very low")
```

Apply this generically to any correlation search with a user field

Sum up the number of events per user. (You can also modify this with severity, risk score, etc.)

Use lookup to add the risk score

If there are multiple offenses, increase risk accordingly. Note that you may want to be careful with actual multiplication as it can create too much noise. See Time Series * First Time Seen Detection

It's often useful to generalize risk as "low" "medium" "high" as it can be more consumable

splunk> .conf2017

# Technique: Risk

## Add this into alert_actions.conf

- Suppose you have this search down exactly how you want it, and now you want to apply it to all your searches, you can easily do this via a macro.
- Then your search becomes:
  […  insert your Correlation Search …]
  | `calculate_risk(user)`
- If you are using ES, you can even build this into the ES Risk Framework by editing the [risk] stanza of:
  $SPLUNK_HOME/etc/apps/SA-ThreatIntelligence/default/alert_actions.conf
- ES Users should also see "Technique: Override Urgency/Severity/Risk" in this doc

# Technique: Subsearches

## Background and Challenges

- "I want to run subsearches that return more than 10k results!"
- "I want to run subsearches that last longer than 60 seconds!"
- "Boy do I like to build my mission-critical detections using this subsearch that returns all of our proxy logs! It even runs way faster as a subsearch!"
  - Hint: that one is a bad one to say!

- Subsearches are very powerful! They can help you build out all kinds of great filters! I assume anyone getting this far probably already knows about subsearches.
- Unfortunately, some don't know that subsearches automatically finalize after 60 seconds (so as far as it gets in 60 seconds is as far as it gets) and can only return a maximum of 10k events. There's solutions (ish) though!

splunk> .conf2017

# Technique: Subsearches

## Returning more than 10k results

- If you have more than 10k results (say you have 15k domains you want to search for) you can use the below. Just keep in mind that there are upper limits - eventually the main search will slow to the point of being unusable if you get to 30k, 40k, fields.
- The secret: if the only value you return from the subsearch is the field "search" then it will be interpreted literally.

```
index=win*security
[ | inputlookup inscope_ad_users.csv
  | stats values(sAMAccountName) as search
  | eval search=
        "(user=" . mvjoin(search, " OR user=") . ")"
]
```

Start with our base dataset, in need of a filter

We now have a list of users

We now have a GIANT single multi-value field

mvjoin now gives us a GIANT single-value field

And now we're back in our search, just with a GIANT list of users

splunk> .conf2017

# Technique: Subsearches

## Taking more than 60 seconds

- Unfortunately there's no magic here, beyond the inherent magic of acceleration.
- There *is* a change you can make to limits.conf, but I've virtually never heard of anyone making that change because it is global across the entire server/cluster.
- So instead, swing down to tstats in the NINJA section, and follow the link to get perspectives on how to approach acceleration!

# Technique: Subsearches

## Working Example

- For a concrete working example, check out the examples under "tstats," and under "stats + eval"
- Notably the example under stats+eval took much much longer than using an "OR" (or multisearch! "Technique: Advanced Commands", towards the end).
- I once did an end-to-end test of performance while looking for threat intel indicators. I compared the performance of doing an [|inputlookup] subsearch to add search criteria, against just looking for all the IPs and then doing a lookup. At 15 indicators, the subsearch was so much faster it was almost silly. At tens of thousands of indicators, the lookup option is faster.

# Advanced Techniques

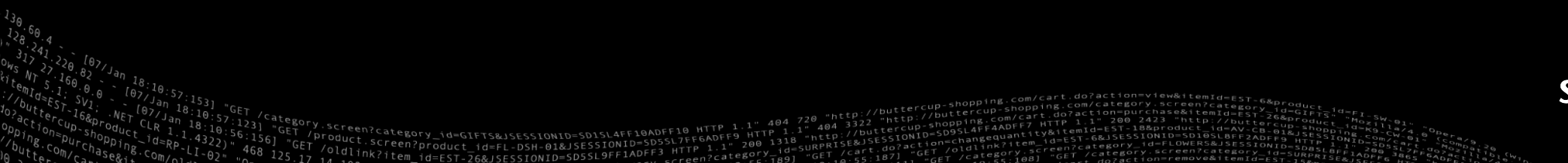
## Let's Get Techy In Here

splunk> .conf2017

# Technique: Summary Indexing

## Background and Challenges

- "I want to look at statistics over one hundred or more days of authentication activity, but the search takes too long to complete"
- "I am analyzing a dataset that requires stats on stats [see this technique elsewhere] but the first stats generates millions of rows and the search is incredibly slow."
- "I have a very slow search using transaction [or threat intel, or etc] and don't want the analysts to have to wait for it, can I just store the results of it?"
- "I need to expose this data to analysts, but they can't see the actual usernames!"
- We will look at each of these scenarios in more depth to explain why Summary Indexing helps here.



# Technique: Summary Indexing

## Essential Data Aggregation

- Take a search, any search, and then index the result, all without license cost!

```
tag=authentication
| stats count as num_auths
        dc(dest) as num_dests
        values(dest) as dest_values
        values(Account_Domain) as Domains
        values(EventCode) as EventCode
        by user,
            Logon_Type,
            action

| collect index=authentication_summaries
```

Start with whatever base search you want

Here we are using stats to pull a number of aggregate metrics. The best part about Summary Indexing is that almost anything you put in front of the "by" clause is free from a performance perspective. More on this in a moment...

Here's the high risk part - whatever we split by increases the number of records (and amount of disk space) exponentially. More on this in a moment…

Now we just pull this data in via collect. What this actually does is write it to a special file in $SPLUNK_HOME/var/spool/splunk

splunk> .conf2017

# Technique: Summary Indexing

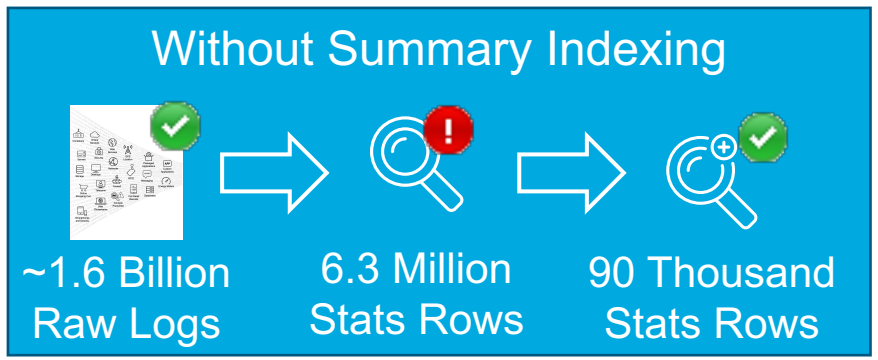
## Summary Indexing for Data Summarization

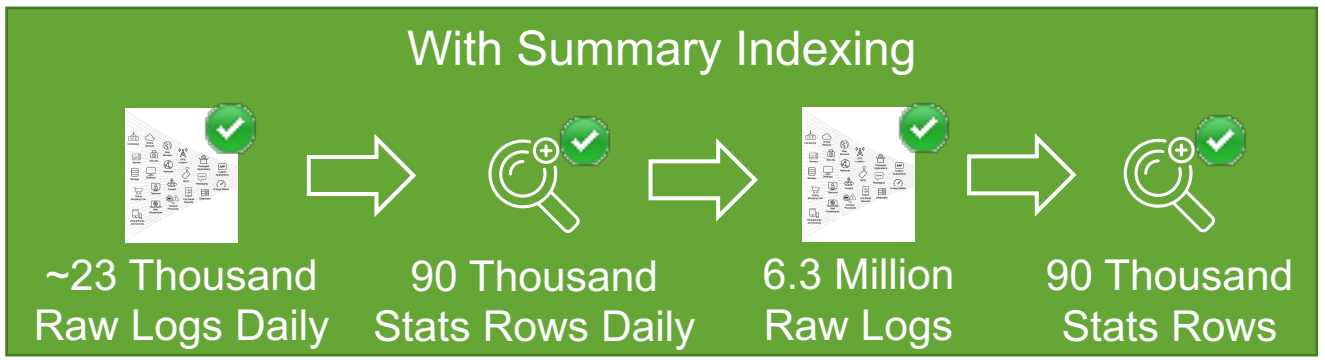
- The example just given with authentication data is a classic illustration of this technique. Suppose you want to analyze a few metrics around Windows Auth data to do behavioral searching on users logging into more searches than normal, but you have 4 TB of Windows Security logs per day, and that would be 120 TB with a 30 day baseline.
- You can use summary indexing to record just the aggregate metrics every day. Suppose somewhat arbitrarily that even including the hostnames, each event was 1 KB. Even in a large org with 300k employees that would consume just 300 MB of disk space per day, and a 30 day baseline search would only have to search 9 GB of data, a reduction of 13,333x.
- We see Summary Indexing being used extensively for scenarios you don’t require the full event fidelity. For example, if you’re indexing huge volumes of Stream, Flow or DNS logs, but sometimes all you need to know if "did these two hosts communicate with each other, and what day/hour" you can summarize the data dramatically.

# Technique: Summary Indexing

## Summary Indexing for Stats on Stats

- As described elsewhere in this presentation, stats on stats is an incredibly powerful technique! The use case we just covered is one example - we would first run stats to prepare our dataset, and then run stats again to actually detect outliers:
  | bucket _time span=1d | stats {metrics} by user, _time | stats {outlier} by user
- There's an innate performance challenge in that, though. Suppose a company with 30k users where you want to detect a change in the number of servers logged into per day, with a 3 month baseline. That would be 90k users * 5 days per week * 14 weeks. That would be 6.3 million rows to keep in memory!
- Splunk has an inherent limit in the amount of rows that can be kept in memory. Above that limit it takes partial result sets and writes them to disk. {This may not be 100% accurate for the internals, but it's generally right:} So maybe you'd get 1M results in memory, and then it would gzip those, and write them to disk. Then it would pull the next 1M results, gzip those and write them to disk, etc. Once it has all the groups of results, it would then read sections of the data back in, decompress them, group them, compress them, and re-write them out to disk, until eventually it is complete.
- That means that a search for 1M row may complete in 4 minutes but a search for 2M rows could complete in 16 minutes. Because the limit here is in MB used, it's not as clear as saying "keep it below 1M rows" but in my experience the slowdown occurs somewhere between 800k and 2M rows depending on what columns you have. Also notably, a 4 minute search taking 16 minutes once per day in the middle of the night isn't actually a problem most of the time, so violating the theshold a little bit is fine. But a 3M row search could then take 30 minutes, 4M row 50 minutes, and eventually your pain becomes great.
- Why so much background discussion? Summary Indexing solves this problem in a lovely fashion. Run the daily aggregation search, where in any day you will only have 90k records (easy). Then when you run the behavioral search, you are looking at raw logs in your summary index and you again only have to track 90k rows.



# Technique: Summary Indexing

## Making Slow Searches Fast

- One of my favorite use cases for both transaction (covered elsewhere in this presentation) and summary indexing is the idea of taking a *very* slow transaction search and then outputting the relevant details into a summary index.
- For example, Ironport logs are a classic use case for transaction, and if you have a 50k employee organization then that search is going to be terribly slow over any long period of time, but the SOC will always want to understand email records.
- | transaction {whatever} | table _time {whatever other fields are relevant to understand} | collect index=our_email_logs
- Then analysts can just run a quick search of index=our_email_logs to get individual pieces. You can still retain the raw data in your Ironport indexes for anything you didn't capture in the summary index.

# Technique: Summary Indexing

## Summary Indexing for Anonymization

- This is generally only reluctantly recommended because of the performance and disk space limitations, but I did want to include it because we are talking about Summary Indexing.
- If you have a data source that you want to expose to a group in your org, but who aren't permitted to see all data (such as employee names), you can summary index your data into a new index.
- index=sensitive | rex mode=sed "s/employee=\"[^\"]*/employee=\"masked" | collect index=masked
- That said, be cautious of trying to do this at really high scale (e.g., limit can vary a lot based on your system, but maybe 150 GB/day?)

# Technique: Summary Indexing

## A Note on Cardinality

- A lot of mechanics in Splunk are dependent on cardinality, which is a measure of how much variability there is in fields. E.g., if you have 30k users and 50k endpoints, | stats … by user would have a maximum of 30k rows, but | stats … by user, dest could theoretically reach 1.5 billion. If you did | stats … by user, dest, EventCode you might end up in the tens of billions.
- This has two implications when it comes to summary indexing. One is why summary indexing helps when doing stats on stats (see a couple of slides ago). The other bigger is when you are choosing what you want to put in your summary index.
- My general recommendation is to put any numbers you might ever need before the by in your stats. For example, when analyzing authentication data, why not track the number of event codes, number of servers, number of logon types, number of Kerberos errors, etc. If I have 30k users and am tracking 7 different metrics, and add an 8th, I see an incremental increase in disk space used, but basically that's it. If I have short field names (remember that we write those to disk, so you pay your storage vendor by the byte), it's almost nothing.
- The flip side, is that I recommend not putting anything after the by clause unless you really need to. Adding "by EventCode" to the end of a Windows Authentication search will increase the number of rows (and amount of disk space by between 6 and 15x depending on how your windows logging is set up.
- I just ran a quick test looking at PAN logs for one hour. In the first example, I don't include app at all. In the second, I include the list of apps. In the third, I split by app.

| KB | # Rows | Search |
|---|---|---|
| 3,160 | 17,584 | index=pan_logs \| stats count dc(dest) as NumDests sum(bytes_*) as sum_bytes_* avg(bytes_*) as avg_bytes_* dc(dest_port) as numDestPorts by src_ip |
| 3,776<br>+ 616 | 17,584 | index=pan_logs \| stats count dc(dest) as NumDests sum(bytes_*) as sum_bytes_* avg(bytes_*) as avg_bytes_* dc(dest_port) as numDestPorts **values(app) as apps** by src_ip |
| 11,700<br>+ 8,540 | 63.239 | index=pan_logs \| stats count dc(dest) as NumDests sum(bytes_*) as sum_bytes_* avg(bytes_*) as avg_bytes_* dc(dest_port) as numDestPorts by src_ip **app** |

# Technique: Summary Indexing

## What about si commands?

- If you have heard of the si commands, my recommendation is that you don't ever use the si commands. They're comparably rigid and difficult to understand.
- If you want to pursue this more, I would recommend the Splunk data Science EDU class.

# Technique: Summary Indexing

## Multiple different summaries in a single index

- One final concept here. The first time you create a summary index, you might put it in a dedicated index, or just use index=summary that ships by default with Splunk.
- When you have 25 different summaries, you will need some way to distinguish them. When you save a summary index via the WebUI, it will ask you if you want to define a marker, which is a kvpair that gets added into the raw event.
- I personally prefer to control my destiny and use the | collect command rather than the WebUI (though probably I should switch to the WebUI). I implement a marker by adding a new field before the | collect.
- index=* | stats … | eval marker="BaselineAuthData" | collect index=xyz
- When defining a marker, you want a medium-long string so that we can use bloom filters and our indexing, but avoid punctuation.

splunk> .conf2017

# Technique: Summary Indexing

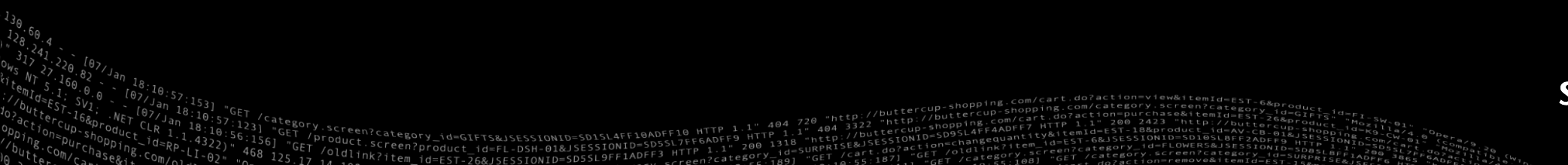
## No Skipped Searches

- While we're here - skipped searches are a common problem on heavily loaded Splunk environments. You want to avoid skipped searches as much as possible, but you can work around that by telling Splunk to use continuous scheduling.
- This setting is in savedsearches.conf, and is called realtime_schedule. (Note, because we want to make this as confusing as possible, a real-time schedule is not the real thing as a real-time search. *I know, I know.*)
- Realtime_schedule defines what happens with a search job is skipped. Either:
  - You skip that time range and move on (bad for summary indexing, and the default)
  - You go wait until you can run for that time range, introducing lag.
- You want the latter (and also to minimize skipped searches by not overloading your Splunk environment).

# Technique: Lookup Caching

## Background and Challenges

- Splunk is a time oriented product, but sometimes we can build better detections with a non-time oriented state store
- "I only want to alert on this if a host has been online for at least a month"
- "I only want to alert if this is the 5th time this has happened in the past month"
- Splunk Security Essentials Use Case:
  - Alert the first time something occurs for any host with a baseline of at least 7 days, and remember the last 90 days.
  - Do this without searching over 90 days every time.



# Technique: Lookup Caching

- You can input a lookup, then output a lookup, and then continue on your search. Run this search every day/hour, and take advantage of a 90 day baseline!

```
tag=authentication
| stats earliest(_time) as earliest
        latest(_time) as latest
        by user, dest
| inputlookup append=t login_tracker.csv
| stats min(earliest) as earliest
        max(latest) as latest
        by user, dest
| where latest > relative_time(now(), "-90d")
| outputlookup sample_cache_group.csv
| where earliest >= relative_time(now(), "-1d@d")
```

Start with whatever base search you want

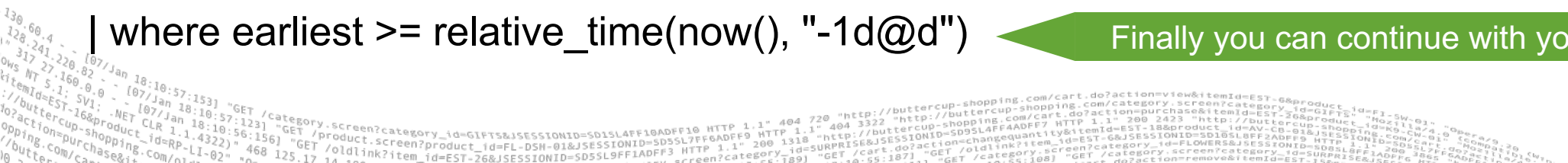
Eventually summarize to a subset of fields that you will be analyzing. Because we want to control the size of the lookup, this should usually be a small number of fields. (More on this next)

*Add* our existing cache with the append=t trigger

Now we can recompute our earliest and latest. The first time was just for our search duration (last day/hour/etc). Now it has the baseline data too.

Now this search is more up to date than our lookup. Update the lookup, and optionally filter out useless data to manage the overall lookup size.

Finally you can continue with your actual detection



splunk> .conf2017

# Technique: Lookup Caching

## How big can your lookup be?

- Pretty big is the general answer. In my head, I try to keep these lookups less than 800 MB, but it can vary depending on how often you run the search itself (e.g., a search every 10 min should be smaller, because otherwise the search won't complete in time.
- The biggest limitation is around disk space and search completion time. If you have 10GB available, don't create big lookups. If you have to read in 8M rows each time the search runs, you won't be able to run it that often.
- Concrete example: first logon by server in a shop with 300k users.
  - Each row: 2 x 10 byte timestamp, username avg 15 bytes, hostname avg 40 bytes = 75 bytes
  - Suppose each user connects to 40 core servers, with 10 random servers per week
  - For each user, that would be 170 servers for a 3 month baseline.
  - 170 servers * 300,000 users * 75 bytes = 3.5 GB - very big! Maybe just track interactive logins.

# Technique: Lookup Caching

## CSV Lookup or kvstore?

- I asked around a bunch when building this technique into Splunk Security Essentials, and basically the answer was "eh, neither is really better."
- Reasons:
  - Because we are writing out the entire list every time (| outputlookup uses append=f) we don't get to take advantage of kvstore incremental update
  - Because we aren't sending to the indexers we don't have to think about the kvstore replication method
  - Because we are doing an | inputlookup append=t instead of | lookup we don't take advantage of kvstore's index capability
- My recommendation: Use CSV lookups
  - There is no benefit to kvstore, and we all know how to manage and deal with csvs. Just way easier.

# Technique: Lookup Caching

## Don’t Big Lookups Hurt Splunk?

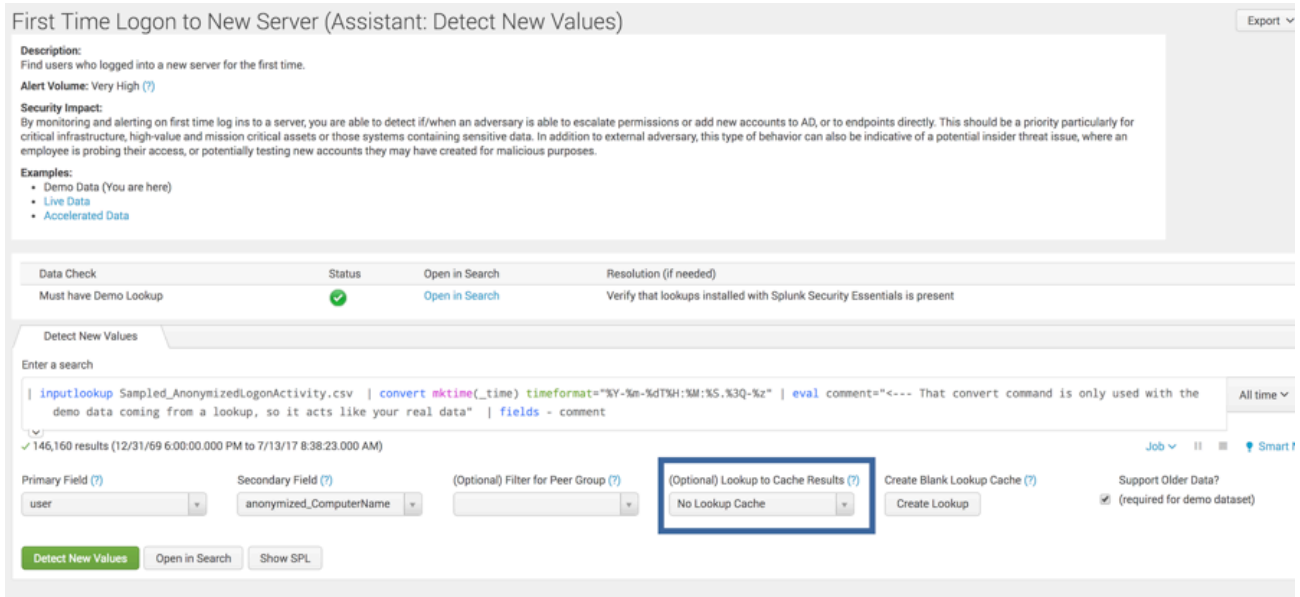
- Biggest Risk to this technique: Lookups use up disk space, and by default will be sent to your indexers in search bundles. If you talk to any large Splunk admin, they will shout you out the door with the terrors of creating 800 MB lookups that break bundle replication.
- Here’s the secret though - this technique doesn’t get any benefit from sending the lookup to the indexers. We aren’t filtering out raw results with it, we’re just using it for enrichment on the Search Head. Keep it out of the bundles via distsearch.conf. This should be common knowledge among advanced admins.
  - https://answers.splunk.com/answers/520843/regex-for-distributed-search-blacklist-not-working.html
  - https://docs.splunk.com/Documentation/Splunk/6.6.2/DistSearch/Limittheknowledgebundlesize
- Now there’s one unknown still here: With SHC, we do still have to replicate those lookups. We don’t have concrete knowledge of where issues lie here, but feedback from our top architects suggest that those limits are way way higher than with normal bundle replication. If you have an 800 MB lookup, it’s probably prudent to not re-generate it every 10 minutes (maybe once per day makes sense? Again, maybe every 10 minutes would be fine though.. We really have no data). I’ve yet to hear about anything short of a >50 GB kvstore breaking SHC replication, which would suggest that we have high limits

# Technique: Lookup Caching

## Working Example

- This has been figured out in Splunk Security Essentials.

1. Download the app off Splunkbase
2. Open up a First Seen Detection (e.g., First Time Logon to New Server)
3. Add a lookup in the "Lookup to Cache Results"
4. Read the description
5. Hit the checkbox and OK
6. Click "Show SPL" to see the SPL

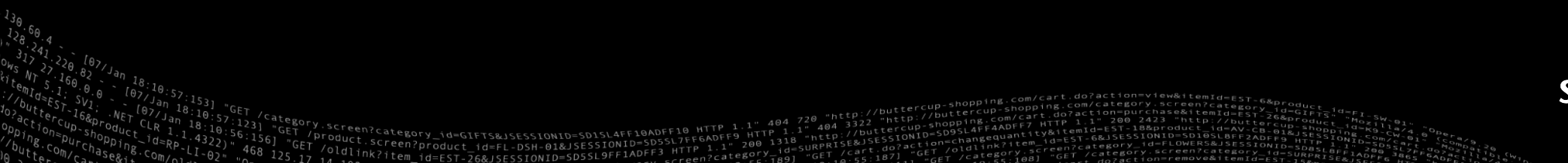



splunk> .conf2017

# Technique: Confidence Checking

## Background and Challenges

- Many times when we look at building use cases, particularly statistical ones, we need to be able to measure the degree to which we have a baseline.
- "I built a first time seen behavioral use case, but it’s alerting on brand new people!"
- "I built a time series analysis behavioral use case, but it’s alerting on someone with only 3 days of baseline!"



# Technique: Confidence Checking

## First Time Seen Detections

- When we do a first time seen detection (see First Time Seen elsewhere in this presentation), we often want to build in confidence checking.

```
tag=authentication
```

Start with whatever base search you want

```
| eval day=strftime(_time, "%d/%m/%Y")
```

This eval gives a single value per day. We could | bucket _time span=1d, rounding the time down, but this lets us keep _time accuracy.

```
| eventstats dc(day) as days_of_baseline by user
```

eventstats is like stats, but without losing the fidelity of the data. This will count up, for each host in the dataset, how many days of data there is.

```
| where days_of_baseline > 7
```

Now we can use where (or search) to filter for hosts where we have enough baseline.

```
| stats earliest(_time) as earliest latest(_time) as latest by user
```

And finally we can continue with our normal first time seen use case, confident that we are including items that just showed up for the first time.

splunk> .conf2017

# Technique: Confidence Checking

## First Time Seen: Do you really want to be confident?

- A great question related to first time seen detection confidence, where you can specify how many days of baseline you expect to see, is "should I do that?" For example, if a new host pops up and starts port scanning your environment, that may very well be a malicious device.
- Ultimately this depends on what your use case is, and it probably makes sense to build out detections specifically targeted to large volumes of new activity. In the following "Variations on First Time Seen" use case we record that activity in a separate index. While I'm not aware of any customer doing this, you could probably build out a detection looking for how many new events there are per user against a role, or what have you.

# Technique: Confidence Checking

## Variations on First Seen Detection

### Check both user and host

- tag=authentication
  | eval day=strftime(_time, "%d/%m/%Y")
  | eventstats dc(day) as days_user by user
  | eventstats dc(day) as days_host by host
  | where days_user > 7 AND days_host > 7
  | stats earliest(_time) as earliest latest(_time) as latest by user
- This allows you to filter out brand new users who log on to many systems, and also brand new hosts (e.g., a new cluster member).

### Tracking new users separately

- tag=authentication
  | eval day=strftime(_time, "%d/%m/%Y")
  | eventstats dc(day) as days_user by user
  | stats earliest(_time) as earliest latest(_time) as latest values(days_user) as days_user by user
  | where earliest > relative_time(now(), "-1d@d")
  | multireport
    [ | where days_user <=7 | collect index=new ]
    [ | where days_user > 7 | collect index=old ]
- This allows you to record new users, but funnel them separately.

splunk> .conf2017

# Technique: Confidence Checking

## Time Series Analysis

- The simplest time series analysis is ensuring you have enough days of baseline to cause the stdev calculation to be meaningful.

```
tag=authentication
| bucket _time span=1d
| stats dc(dest) as count by user, _time
| stats count as num_data_samples
        max(eval(if(_time >= relative_time(now(),
            "-1d@d"), count,null))) as latest
        avg(eval(if(_time<relative_time(now(),
            "-1d@d"), count,null))) as avg
        stdev(eval(if(_time<relative_time(now(),
            "-1d@d"), count,null))) as stdev
    by user
| where latest > avg + stdev * 3 AND
          num_data_samples > 7
```

Start with whatever base search you want

This is the standard time series behavioral detection use case. But note the count as num_data_samples - because that is coming after the stats … by user _time, this will count how many days we end up with, for each user. If a user only has a few data points, standard deviation is a worthless data point. In some scenarios, you would even want to have at least 20 or 30 data points.

In the same breath that we track the average and standard deviations, we can also filter out users that don't have enough days of baseline.

splunk> .conf2017

# Technique: Confidence Checking

## Time Series Analysis

- Based on observed behavior, standard deviation tends to work the best when you have a use case that naturally has some deviation. For example, number of logon messages per day, or number of pages printed. Usually those will differ from day to day.
- You're more likely to see what people think of as noise or false positives in datasets with limited variation, such as number of systems interactively logged into per day (e.g., by sitting at the computer or via remote desktop), or if you have a user who just prints one page now and again. Most users just log into one system per day, so their average is 1, and their stdev is going to be 0 or very close to it, so even setting 6 stdevs above the average might alert for 2 systems.
- You might decide that you care if someone who usually just logs into one system logs into a second, even if it's likely benign. But you probably don't care about someone who usually prints one page, suddenly printing two pages.
- There are two approaches I have most frequently seen to manage this:
  - Static Filters - don't alert if the # is less than X, or the # increase is less than X.
  - Relative Filters - only alert if this is 2x their average AND 3 stdev above the average.

# Technique: Confidence Checking

## Variations on Time Series

### Adding Static Filters

- tag=authentication | bucket _time span=1d
  | stats dc(dest) as count by user, _time
  | stats count as num_data_samples
  max(eval(if(_time >= relative_time(now(),
  "-1d@d"), count,null))) as latest
  avg(eval(if(_time<relative_time(now(),
  "-1d@d"), count,null))) as avg
  stdev(eval(if(_time<relative_time(now(),
  "-1d@d"), count,null))) as stdev
  by user
  | where latest > avg + stdev * 3 AND
  num_data_samples > 7 AND
  latest > 5 AND (latest - avg) > 5

### Adding Relative Filters

- tag=authentication | bucket _time span=1d
  | stats dc(dest) as count by user, _time
  | stats count as num_data_samples
  max(eval(if(_time >= relative_time(now(),
  "-1d@d"), count,null))) as latest
  avg(eval(if(_time<relative_time(now(),
  "-1d@d"), count,null))) as avg
  stdev(eval(if(_time<relative_time(now(),
  "-1d@d"), count,null))) as stdev
  by user
  | where latest > avg + stdev * 3 AND
  num_data_samples > 7 AND
  latest > avg * 2

splunk> .conf2017

# Technique: Confidence Checking

## Time Series Analysis Variation: Signal to Noise Ratio

- The simplest time series analysis is ensuring you have enough days of baseline to cause the stdev calculation to be meaningful.

```
tag=authentication
| bucket _time span=1d
| stats dc(dest) as count by user, _time
| stats count as num_data_samples
        max(eval(if(_time >= relative_time(now(),
            "-1d@d"), count,null))) as latest
        avg(eval(if(_time<relative_time(now(),
            "-1d@d"), count,null))) as avg
        stdev(eval(if(_time<relative_time(now(),
            "-1d@d"), count,null))) as stdev
    by user
| where latest > avg + stdev * 3 AND
        num_data_samples > 7
```

Start with whatever base search you want

This is the standard time series behavioral detection use case. But note the count as num_data_samples - because that is coming after the stats … by user _time, this will count how many days we end up with, for each user. If a user only has a few data points, standard deviation is a worthless data point. In some scenarios, you would even want to have at least 20 or 30 data points.

In the same breath that we track the average and standard deviations, we can also filter out users that don't have enough days of baseline.

splunk> .conf2017

# Technique: Managing Alert Fatigue

## Background and Challenges

- One of the biggest inherent challenges of the modern security world is alert fatigue. The sheer volume of security alerts that we experience dooms so many SOCs, and we evidence of these problems in many breach debriefs.
- Fortunately, there are several techniques for dealing with this inside of a Splunk world.

# Technique: Managing Alert Fatigue

## Using Risk to aggregate alerts

- If you have low confidence alerts, send them just into the risk index in ES (or build your own -- | eval risk_object=src_ip | collect index=risk) and aggregate.

```
index=risk earliest=-30d | stats values(source) as search_names sum(risk_score)
as thirty_day_risk sum(eval(if(_time > relative_time(now(), "-1d"),risk_score,0))) as
one_day_risk by risk_object | eval threshold_1day = 500, threshold_30day = 1200
| eventstats avg(thirty_day_risk) as avg_thirty_day_risk stdev(thirty_day_risk) as
stdev_thirty_day_risk

| where one_day_risk>threshold_1day OR thirty_day_risk>threshold_30day OR
thirty_day_risk>(avg_thirty_day_risk + 3 * stdev_thirty_day_risk)

| eval risk_score_reason = case(one_day_risk>threshold_1day, "One Day Risk
Score above " . threshold_1day, thirty_day_risk>threshold_30day . " on " .
strftime(now(), "%m-%d-%Y"), "Thirty Day Risk Score above " . threshold_30day,
1=1, "Thirty Day Risk Score more than three standard deviations above normal (>"
. round((avg_thirty_day_risk + 3 * stdev_thirty_day_risk),2) . ")") | fields - avg*
stdev*

| table risk_object risk_score one_day_risk thirty_day_risk risk_score_reason
```

See a full description of this search under the "Multi-Scenario Alerts" and "Inline Comments" sections

splunk> .conf2017

# Technique: Managing Alert Fatigue

## Using Statistics to Manage Fatigue

- Similar to the risk approach, even in your normal ticketing flow you can take high priority alerts and bring them to the top of the list by creating meta-notables.

```
tag=ids tag=attack
| bucket _time span=1d
| stats count by severity signature dest _time
| stats sum(count) as count
        avg(count) as avg
        stdev(count) as stdev
        sum(eval(if(_time > relative_time(now(),  "-1d"),
            count, 0))) as recent_count
        min(_time) as earliest
    by severity signature dest
| eventstats avg(avg) as avg_num_per_dest
                avg(earliest) as avg_earliest
                sum(count) as sig_wide_count
                sum(recent_count) as sig_wide_recent_count
   by signature
 | where NOT (avg_earliest < relative_time(now(), "-1y") AND
            sig_wide_recent_count / sig_wide_recent_count < 0.05 AND
            priority <=3)
```

Start by building up a set of aggregate statistics for our dataset.

Use eventstats to add additional context, in this case about the IDS Signature

We now have a large body of fields with relevant data about this event. Use | where to apply your logic about what you do or don't want to see.

splunk> .conf2017

# Technique: Managing Alert Fatigue

## Build Specific Application Logic

- Similar to the risk approach, even in your normal ticketing flow you can take high priority alerts and bring them to the top of the list by creating meta-notables.
- Simple Example:
- index=notable
  | stats dc(search_name) as NumRules
  values(search_name)
  by dest
  | where NumRules>2
- More Specific Example:
- (index=notable Antivirus OR ids) OR (index=proxy category="")
  | eval dest=case(index="proxy", src, index="notable", dest)
  | stats dc(search_name) as NumRules
  count(eval(index="proxy")) as NumUncategorizedHits
  by dest
  | where NumRules>1 AND NumUncategorizedHits > 0

splunk> .conf2017

# Technique: Managing Alert Fatigue

## Increase Logging

- If you have a mundane alert (e.g., low severity IDS alert, AV successful clean, etc.), why not increase logging on that host for a while?
- With ES, you can use Stream to do network capture, or leverage any other adaptive response actions. With or without ES, you can use your EDR solution. Many customers leverage the Palo Alto Networks app or expect scripts to add suspect hosts to groups that have additional logging. Etc.
- Write additional correlation rules based on that increased logging to look for higher confidence, higher severity alerts.

# Technique: Managing Alert Fatigue

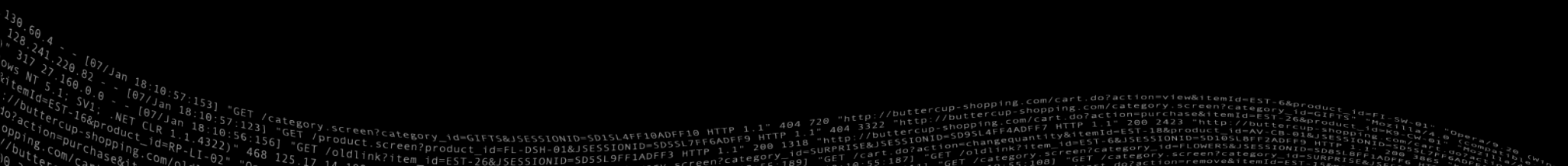
## Leverage Machine Learning

- With Machine Learning, you can build extremely powerful models and techniques for finding outliers programmatically.
- Look at Splunk UBA - this is what they do.
- Look at the ML Toolkit App

# Technique: Transaction

## Background and Challenges

- Being able to group events that are similar is super important.
- Transaction has a terrible reputation for being slow, because it is slow. But there are scenarios where it is super easy, and we all work in technology because we like to go against conventional wisdom, right?
  - Use Transaction for very low event volumes, e.g.: sourcetype=win*security super.exe rare.exe executables.exe | transaction host maxpause=10m maxspan=5h
  - Use Transaction asynchronously to populate a summary index, e.g.: sourcetype=ironport OR sourcetype=cisco:esa | transaction MID | fields - _raw | fields {…} | collect index=our_email
- Transaction doesn’t fail on by the hour borders (| bucket _time span=1h | stats whatever by _time)



# Technique: Transaction

## Transaction for low event volume

- When you can filter your incoming event flow to a low volume, even if transaction is 10x slower, who cares?

```
sourcetype=win*security EventCode=4688
[| inputlookup suspicious_processes.csv]
```

We know that transaction is slow, so the key here is using it for a dataset where you won't send much data to transaction.

```
| transaction host
maxpause=10m
maxspan=10h
```

Now we can use transaction!

# Technique: Transaction

## Transaction with Summary Indexing

- Sometimes transaction is *way* easier. Like, "mere mortals don't have the SPL Skill to use stats + eventstats + streamstats + whatever magic allows you to see your way to the desired result. In this case, embrace the slow. Use transaction asynchronously, and then send the results to a summary index. To help avoid skipped searches, use realtime_schedule=1 in savedsearches.conf. (Check Summary Indexing in this doc.)

```
sourcetype=ironport OR sourcetype=cisco:esa
```

Ironport logs are not the only good transaction example, but they're certainly the classical example of something really hard to do without transaction

```
| transaction MID ICID …
maxpause=5m
maxspan=1h
```

Now we can use transaction

```
| table _time mid icid dcid recipient sender…
| collect index=our_email
```

Now you can send the completed results into your summary indexes. Note that you have to make sure you're not skipping this search.

splunk> .conf2017

# Technique: Transaction

## Speeding up a slow-ish Transaction

- Transaction has many options that are often ignored, to the peril of the operator.
- Go check out the docs page for Transaction.. I'll wait: http://docs.splunk.com/Documentation/Splunk/latest/SearchReference/transaction
- See all those maxspan, maxpause, max etc? Those are to control the amount of memory used by transaction. Splunk controls how much data can exist simultaneously in memory, and when it hits that limit it writes out things to disk.
- More memory in use, less speed in search.
- If you have a larger search with transaction (e.g., your email logs), never run it without maxpause, maxspan controls.

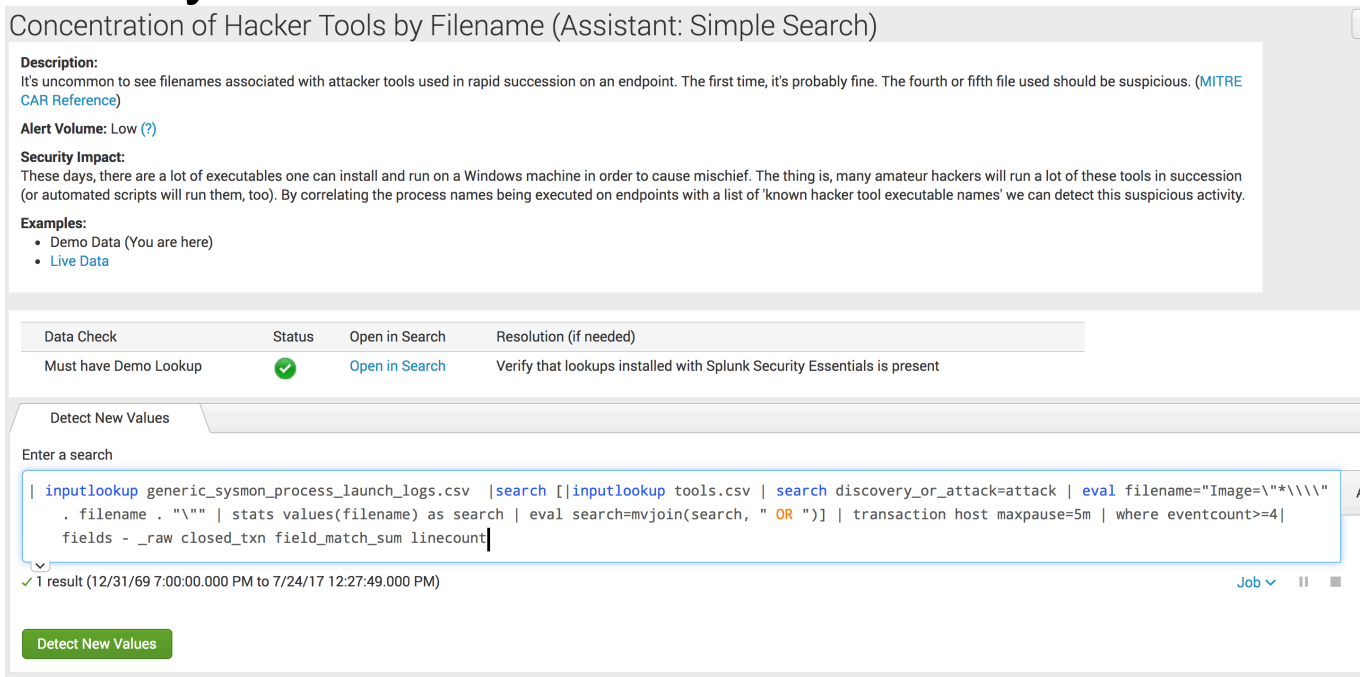
# Technique: Transaction

## Working Example

- In Splunk Security Essentials we have any example using a small volume of logs

1. Download the app off Splunkbase
2. Open up Concentration of Hacker Tools by Filename
3. Click "Show SPL" to see the SPL

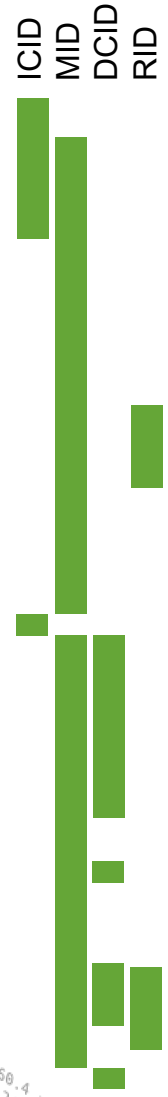
- This search pulls in only process launches for suspicious attacker tools, and limits to just 5 minutes
- If this search is excessively slow, you either have a very odd network or are terrifyingly compromised




splunk> .conf2017

# Technique: Transaction

## Cisco ESA Logs (AKA Ironport Logs) - the log files themselves




```
Wed Feb 12 19:48:37 2014 Info: New SMTP ICID 1000000201 interface Data2 (192.0.2.44) address 203.0.113.83 reverse dns host unknown verified no
Wed Feb 12 19:48:37 2014 Info: ICID 1000000201 ACCEPT SG UNKNOWNLIST match sbrs[0.0:10.0] SBRS 5.1
Wed Feb 12 19:48:37 2014 Info: ICID 1000000201 TLS success protocol TLSv1 cipher AES128-SHA
Wed Feb 12 19:48:37 2014 Info: Start MID 500000014 ICID 1000000201
Wed Feb 12 19:48:37 2014 Info: MID 500000014 ICID 1000000201 From:
Wed Feb 12 19:48:37 2014 Info: MID 500000014 ICID 1000000201 RID 0 To:
Wed Feb 12 19:48:37 2014 Info: MID 500000014 ICID 1000000201 RID 1 To:
Wed Feb 12 19:48:38 2014 Info: MID 500000014 Message-ID ''
Wed Feb 12 19:48:38 2014 Info: MID 500000014 Subject 'FW: Some Subject Matter'
Wed Feb 12 19:48:38 2014 Info: MID 500000014 ready 52076 bytes from
Wed Feb 12 19:48:38 2014 Info: LDAP: Masquerade query Rewrites-Inbound MID 500000014 address bill.jones@example.com to bill.jones@example.com
Wed Feb 12 19:48:38 2014 Info: LDAP: Masquerade query Rewrites-Inbound MID 500000014 address reginald.brown@example.com to reginald.brown@example.com
Wed Feb 12 19:48:38 2014 Info: LDAP: Masquerade query Rewrites-Inbound MID 500000014 address amy.johnson@example.com to amy.johnson@example.com
Wed Feb 12 19:48:38 2014 Info: MID 500000014 rewritten to MID 500000031 by LDAP rewrite
Wed Feb 12 19:48:38 2014 Info: MID 500000031 ICID 0 From:
Wed Feb 12 19:48:38 2014 Info: LDAP: Reroute query AD.routing MID 500000014 RID 0 address bill.jones@example.com to [('bill.jones@internal.example.com', '')]
Wed Feb 12 19:48:38 2014 Info: MID 500000031 ICID 0 RID 0 To:
Wed Feb 12 19:48:38 2014 Info: LDAP: Reroute query AD.routing MID 500000014 RID 1 address amy.johnson@example.com to [('amy.johnson@internal.example.com', '')]
Wed Feb 12 19:48:38 2014 Info: MID 500000031 ICID 0 RID 1 To:
Wed Feb 12 19:48:38 2014 Info: Message finished MID 500000014 done
Wed Feb 12 19:48:38 2014 Info: MID 500000031 attachment 'image001.jpg'
Wed Feb 12 19:48:38 2014 Info: MID 500000031 attachment 'image002.jpg'
Wed Feb 12 19:48:38 2014 Info: MID 500000031 Custom Log Entry: Attachment Names: image001.jpg, image002.jpg
Wed Feb 12 19:48:38 2014 Info: MID 500000031 Custom Log Entry: Attachment Sizes: 1736, 1525
Wed Feb 12 19:48:38 2014 Info: MID 500000031 Custom Log Entry: Attachment Types: image/jpeg, image/jpeg
Wed Feb 12 19:48:38 2014 Info: ICID 1000000201 close
Wed Feb 12 19:48:38 2014 Info: New SMTP DCID 70000094 interface 192.0.2.44 address 192.0.2.89 port 25
Wed Feb 12 19:48:38 2014 Info: DCID 70000094 STARTTLS command not supported
Wed Feb 12 19:48:38 2014 Info: Delivery start DCID 70000094 MID 500000027 to RID [0]
Wed Feb 12 19:48:38 2014 Info: Message done DCID 70000094 MID 500000027 to RID [0]
Wed Feb 12 19:48:38 2014 Info: MID 500000031 matched all recipients for per-recipient policy DEFAULT in the inbound table
Wed Feb 12 19:48:39 2014 Info: Delivery start DCID 70000094 MID 500000026 to RID [0]
Wed Feb 12 19:48:39 2014 Info: Message done DCID 70000094 MID 500000026 to RID [0]
Wed Feb 12 19:48:39 2014 Info: Delivery start DCID 70000094 MID 500000033 to RID [0]
Wed Feb 12 19:48:39 2014 Info: Message done DCID 70000094 MID 500000033 to RID [0]
Wed Feb 12 19:48:40 2014 Info: MID 500000031 interim verdict using engine: CASE spam negative
Wed Feb 12 19:48:40 2014 Info: MID 500000031 using engine: CASE spam negative
Wed Feb 12 19:48:40 2014 Info: Delivery start DCID 70000094 MID 500000049 to RID [0]
Wed Feb 12 19:48:40 2014 Info: MID 500000031 interim AV verdict using Sophos CLEAN
Wed Feb 12 19:48:40 2014 Info: MID 500000031 antivirus negative
Wed Feb 12 19:48:40 2014 Info: MID 500000031 Outbreak Filters: verdict negative
Wed Feb 12 19:48:40 2014 Info: MID 500000031 queued for delivery
Wed Feb 12 19:48:40 2014 Info: Message done DCID 70000094 MID 500000049 to RID [0]
Wed Feb 12 19:48:40 2014 Info: Delivery start DCID 70000094 MID 500000031 to RID [0, 1]
Wed Feb 12 19:48:40 2014 Info: Message done DCID 70000094 MID 500000031 to RID [0, 1]
Wed Feb 12 19:48:40 2014 Info: MID 500000031 RID [0, 1] Response '2.6.0  Queued mail for delivery'
Wed Feb 12 19:48:40 2014 Info: Message finished MID 500000031 done
Wed Feb 12 19:48:40 2014 Info: DCID 70000094 close
```

- ICID - Incoming Connection ID, can contain many MIDs
- MID - Message ID
- DCID - Destination Connect ID, can contain many MIDs
- RID - Recipient ID (many Recipients per MID, though MID present in all lines)

splunk> .conf2017

# Technique: Transaction

## Cisco ESA Logs (AKA Ironport Logs) - transaction

- With transaction, we need to jump through a one small hoop to get the relevant details from the ICID in every MID. This is because the ICID contains a few general parameters (SSL version, src_ip, etc.) that we want noted for every MID

```
sourcetype=ironport OR sourcetype=cisco:esa

| eventstats values(TLS) as TLS values(src_ip)
as src_ip values(…) as … by ICID

| transaction MID
maxpause=5m
maxspan=1h

| fields - _raw | fields sender recipient src_ip TLS
```

There is at least one message with ICID and MID in it, so if we use eventstats to distribute the important values to everything with an ICID, we will be able to just use transaction on MID and have what we want.

Now we can use transaction

Now you can send the completed results into your summary indexes. Note that you have to make sure you're not skipping this search.

Test Environment, over 3 hours of data (23k messages): 119 seconds

# Technique: Transaction

## Cisco ESA Logs (AKA Ironport Logs) - eventstats, stats

- Without transaction, you just leverage stats to group things together. You will often need to do streamstats or some other cleverness (defining day with strptime) to get around re-use of IDs, though Cisco ESA is simpler.

```
sourcetype=ironport OR sourcetype=cisco:esa

| eventstats values(TLS) as TLS values(src_ip)
as src_ip values(…) as … by ICID

| stats values(icid) AS icid
        values(src*) AS src*
    by mid

| eval recipient_count=mvcount(recipient)
```

There is at least one message with ICID and MID in it, so if we use eventstats to distribute the important values to everything with an ICID, we will be able to just use transaction on MID and have what we want.

Here we can use stats instead of transaction for a 3x speed boost!

Test Environment, over 3 hours of data (23k messages): 40 seconds - 3x faster

# Technique: Transaction

## Fuller Example of Grabbing Complete Ironport Logs

- This is a more complete example that can be run every 5 min

```
sourcetype=cisco:esa* earliest=-20m

| eventstats values(sending_server) as sending_server values(sending_server_dns_status) as
sending_server_dns_status values(sending_server_dkim) as sending_server_dkim
values(sending_server_tls_status) as sending_server_tls_status
values(sending_server_tls_cipher) as sending_server_tls_cipher
values(sending_server_whitelist) as sending_server_whitelist by icid

| stats min(_time) as _time max(_time) as email_processing_complete_time
count(eval(searchmatch("Message Finished MID"))) as complete_count
count(eval(searchmatch("Start MID"))) as start_count values(d) as d values(message_id) as
message_id values(message_subject) as message_subject values(mid) as mid
values(recipient) as recipient values(sender) as sender values(spam_status) as spam_status
values(encoding) as encoding values(subject) as subject values(attachment) as attachment
values(queue) as queue values(message_scan_error) as message_scan_error
values(message_size) as message_size values(sending_server) as sending_server
values(sending_server_dns_status) as sending_server_dns_status
values(sending_server_dkim) as sending_server_dkim values(sending_server_tls_status) as
sending_server_tls_status values(sending_server_tls_cipher) as sending_server_tls_cipher
values(sending_server_whitelist) as sending_server_whitelist values(icid) as icid values(dcid)
as dcid by mid

| where complete_count > 1 AND start_count > 1 AND
email_processing_complete_time >= relative_time(now(), "-7m@m") AND
email_processing_complete_time < relative_time(now(), "-2m@m")

| collect index=parsed_emails
```

For our base dataset we can pull in all the data from the last 20 minutes

Eventstats to pull fields out of the icid, and connect them with the MID

Now we can pull all the fields we care about

This where statement looks for complete transactions (both start and complete messages), that are in a five minute window

Finally, send results into the summary index

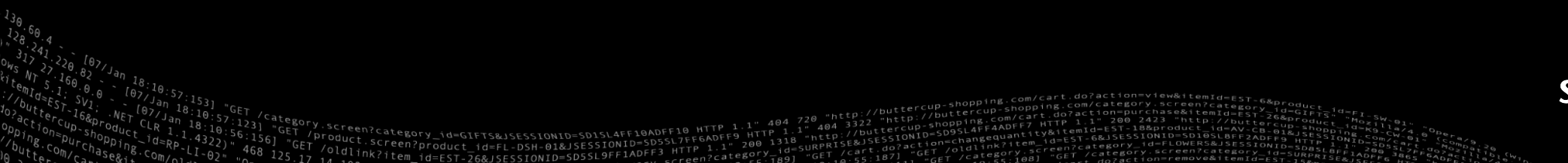
splunk> .conf2017

# Technique: First Time Seen Detection

## Background and Challenges

- "I want to know the first time someone connects a USB drive"
- "I want to know when someone first prints"
- "I want to know the first time {someone/something} {does / sees anything}"

- First Time Seen detections are very powerful! They are the cornerstone for many simpler UEBA tools, and they've been done with Splunk Enterprise / Enterprise Security for ages.
- Splunk Security Essentials showcases many examples of these detections, but the possibilities are almost literally limitless, and driven primarily by your data and your use cases.



# Technique: First Time Seen Detection

## Your first First Time Seen detection - First Logon to New Server

- When doing a first time seen detection, you just need to leverage stats earliest() and stats (latest)!

```
sourcetype=win*security
```

Start with our Windows logs (maybe filter for just logon activity - I grabbed a list of Windows Logon Event IDs for Splunk Security Essentials)

```
| stats earliest(_time) as earliest latest(_time) as latest  by user, dest
```

We can now just use stats earliest and latest to find the most recent logon time - technically we don't need the latest, but it's pretty cheap and useful

```
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)
```

Finally we check to see if the earliest time (first logon) is in the last day.

That easy!

# Technique: First Time Seen Detection

## First Time Seen with Peer Group!

- Peer group detection complicates the query - see multireport under Advanced Commands, later in this doc.

```
sourcetype=win*security | lookup peer_group user OUTPUT peergroup | makemv
peergroup delim=","

| multireport
   [| stats values(*) as * by user dest]
   [| stats values(eval(if(earliest>=relative_time(maxlatest,"-1d@d"),dest,null))) as
peertoday values(eval(if(earliest<relative_time(maxlatest,"-1d@d"),dest,null))) as
peerpast by peergroup dest]

| eval user=coalesce(user, peergroup) | fields - peergroup | stats values(*) as * by
user  dest

| where isnotnull(earliest)


| eval isOutlier= if(isnotnull(earliest) AND earliest>=relative_time(maxlatest,"-
1d@d") AND isnull(peerpast),1,0)
```

Start with our Windows Logon events

Use multireport to pull the earliest and latest both for the user, and for their peer group

Then consolidate all the values for each user

A quirk of the peer group analysis is that we can end up with users in the peer group who have never logged into that host - let's filter them out

Finally, we look for the user's earliest and latest, where the peergroup past is empty

Get the latest and greatest for this detection with Splunk Security Essentials

splunk> .conf2017

# Technique: First Time Seen Detection

## First Time Seen with a Lookup Cache!

- By using a lookup cache, we don't have to look over 30,60,100 days of data every time you run the search.

```
sourcetype=win*security
| stats earliest(_time) as earliest latest(_time) as latest  by user, dest
| inputlookup append=t lookup_cache.csv
| stats min(earliest) as earliest max(latest) as latest by  user, dest
| outputlookup lookup_cache.csv
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)
```

I'm not even going to explain this here! Go to the "Technique: Lookup Caching" later in the doc

While we're at it, you should go check "Technique: Confidence Checking" later in the doc

If you like this, you'll love "Technique: Time Series Detection" up next!

Get the latest and greatest for this detection with Splunk Security Essentials

splunk> .conf2017

# Technique: First Time Seen Detection

## First Time Seen with a Lookup Cache AND Peer Group! You're a mad man!

- Just for fun, because I'm just copy-pasting from Splunk Security Essentials at this point - First Logon to New Server with both a Peer Group AND a Lookup Cache!

```
sourcetype=win*security
| stats earliest(_time) as earliest latest(_time) as latest  by user, dest
| inputlookup append=t sample_cache_group.csv | stats min(earliest) as earliest
max(latest) as latest by  user, dest
| outputlookup sample_cache_group.csv
| lookup peer_group.csv user OUTPUT peergroup | makemv peergroup delim=","
| multireport [| stats values(*) as * by user  dest  ] [| stats
values(eval(if(earliest>=relative_time(now(),"-1d@d"),dest ,null))) as peertoday
values(eval(if(earliest<relative_time(now(),"-1d@d"),dest ,null))) as peerpast by
peergroup dest   ]
| eval user=coalesce(user, peergroup) | fields - peergroup | stats values(*) as * by
user  dest
| where isnotnull(earliest)
| isOutlier= if(isnotnull(earliest) AND earliest>=relative_time(now(),"-1d@d") AND
isnull(peerpast),1,0)
```

Multi-Report AND a mid-search outputlookup? High Five time.

Get the latest and greatest for this detection with Splunk Security Essentials

splunk> .conf2017

# Technique: First Time Seen Detection

## This is Very Generic

- While we've been having fun with First Logon to New Server, this same search works for any first time seen detection

First Logon to New Server
sourcetype=win*security
| stats earliest(_time) as earliest latest(_time) as latest by user, dest
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)

Authentication against a New Domain Controller
sourcetype=win*security
| stats earliest(_time) as earliest latest(_time) as latest by user, dc
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)

First Access to a New Source Code Repository
sourcetype=source_code_access
| stats earliest(_time) as earliest latest(_time) as latest by user, repo
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)

First External Email Claiming to be Internal from Server
sourcetype=cisco:esa src_user=*@mycompany.com src!=10.0.0.0/8
| stats earliest(_time) as earliest latest(_time) as latest by user, src
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)

Familiar Filename on a New Path
Sourcetype=win*security EventCode=4688 `IncludeMicrosoftFiles`
| stats earliest(_time) as earliest latest(_time) as latest by filename, path
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)

New Database Table Accessed
sourcetype=database
| stats earliest(_time) as earliest latest(_time) as latest by user, table
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)

New Interactive Logon by Service Account
sourcetype=win*security user=srv_* Logon_Type=2 OR .. 11 .. 12
| stats earliest(_time) as earliest latest(_time) as latest by user, dest
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)

New Parent Process for cmd.exe
sourcetype=win*security EventCode=4688 filename=4688
| stats earliest(_time) as earliest latest(_time) as latest by parent_process
| eval isOutlier=if(earliest >= relative_time(now(), "-1d@d"), 1, 0)

splunk> .conf2017

# Technique: Time Series Detection

## Background and Challenges

- "I want to detect someone who {prints more / logs in more / logs into more devices / anything more} than usual"
- Time series analytics are very powerful! They are the cornerstone for many simpler UEBA tools, and they've been done with Splunk Enterprise / Enterprise Security for ages.
- Splunk Security Essentials showcases many examples of these detections, but the possibilities are almost literally limitless, and driven primarily by your data and your use cases.

# Technique: Time Series Detection

## Trends among Splunk Core/ES Use Cases

- Frequently:
  - Simple - medium complexity security logic
  - Time series oriented
  - Often require tuning
- Rarely:
  - Advanced Machine Learning driven
  - Manages state in a non-time series fashion
  - Normalizes identities based on DHCP



115

# Technique: Time Series Detection

## Example UBA Splunk Core/ES Use Cases

- Anything traditionally rules built, e.g. first logon to new system
- Unusually high # of connection attempts
- Unusually high # of records accessed / printed / exported / etc
- Unusually high # of files changed
- Rare SHAs, TLS Certs, etc.
- User actions from service account (Proxy, Failed Password Changes)
- User actions from expired account


116

# Technique: Time Series Detection

## Lateral Movement

- We often want to each for attackers who are expanding their systems controlled, and data accessed.
- One technique for this is looking at # of logins per user, or # of destinations per source IP
- Network Data provides source of truth
  - I usually talk to 10 hosts
  - Then one day I talk to 10,000 hosts
  - ALARM!
- How would we approach that? By doing a time series analysis.

# Technique: Time Series Detection

## Detecting Variations Visually

```
| stats count sparkline(dc(dest)) by src_ip
```

Consistently large

| src_ip | count | dc(dest) | sparkline(dc(dest_ip)) |
|---|---|---|---|
| 10.78.113.24 | 60168 | 10699 |  |

Inconsistent!

| src_ip | count | dc(dest) | sparkline(dc(dest_ip)) |
|---|---|---|---|
| 10.174.30.148 | 2219 | 210 |  |

# Technique: Time Series Detection

## What is Standard Deviation?

- A measure of the variance for a series of numbers

| User | Day One | Day Two | Day Three | Day Four | Avg | Stdev |
|---|---|---|---|---|---|---|
| Jane | 100 | 123 | 79 | 145 | 111.75 | 28.53 |
| Jack | 100 | 342 | 3 | 2 | 111.75 | 160.23 |

| User | Day Five | # StDev Away from Average … aka How Unusual? |
|---|---|---|
| Jane | 500 | **12.6** |
| Jack | 500 | 2.42 |

# Technique: Time Series Detection

## Make it a Better Correlation Search

```
… | stats avg( eval(
        if(_time < relative_time(now(),   "-1d@d"),
            count, null)
    )) as average …
```

- Exclude Yesterday's Value using Stats + Eval so your avg and stdev are accurate
- This is as hard as it gets

| User | Day One | Day Two | Day Three | Day Four | Avg | Stdev |
|---|---|---|---|---|---|---|
| Jane | 100 | 123 | 79 | 145 | 111.75 | 28.53 |
| Jack | 100 | 342 | 3 | 2 | 111.75 | 160.23 |

| User | Day Five | # StDev Away from Average … aka How Unusual? |
|---|---|---|
| Jane | 500 | **12.6** |
| Jack | 500 | 2.42 |



splunk> .conf2017

# Technique: Time Series Detection

## Correlation Search Version

- sourcetype="pan:traffic"
- | bin span=1d _time | stats dc(dest) as count by src _time
- | stats max(eval(if(_time >= relative_time(now(), "1d"), count, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d"),count,null))) as average, stdev(eval(if(_time < relative_time(now(), "-1d"),count,null))) as stdev by src
- | where latest>stdev+average

Maybe user instead?

Maybe 2*stdev instead?



splunk> .conf2017

# Technique: Time Series Detection

## Other Variations: # of Logins Per Day

- index=windows OR index=login user=*
- | bin span=1d _time | stats count by user _time
- | stats max(eval(if(_time >= relative_time(now(), "1d"), count, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d"),count,null))) as average, stdev(eval(if(_time < relative_time(now(), "-1d"),count,null))) as stdev by user
- | where latest>stdev+average

# Technique: Time Series Detection

## Other Variations: # of Servers Logged Into

- index=windows OR index=login user=*
- | bin span=1d _time | stats dc(host) as count by user _time
- | stats max(eval(if(_time >= relative_time(now(), "1d"), count, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d"),count,null))) as average, stdev(eval(if(_time < relative_time(now(), "-1d"),count,null))) as stdev by user
- | where latest>stdev+average

# Technique: Time Series Detection

## Other Variations: # of pages printed

- index=windows pages printed
- | bin span=1d _time | stats sum(Num_Pages) as count by user _time
- | stats max(eval(if(_time >= relative_time(now(), "1d"), count, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d"),count,null))) as average, stdev(eval(if(_time < relative_time(now(), "-1d"),count,null))) as stdev by user
- | where latest>stdev+average

# Technique: Time Series Detection

## Other Variations: # of Credit Cards Viewed

- index=crm_logs viewed card
- | bin span=1d _time | stats dc(card_id) as count by user _time
- | stats max(eval(if(_time >= relative_time(now(), "1d"), count, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d"),count,null))) as average, stdev(eval(if(_time < relative_time(now(), "-1d"),count,null))) as stdev by user
- | where latest>stdev+average

# Technique: Time Series Detection

## Other Variations: # of Files Written to USB

- index=sep* api="File Write"  tag=target_users `sep_write_exclude`
- | bin span=1d _time | stats count by user _time
- | stats max(eval(if(_time >= relative_time(now(), "1d"), count, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d"),count,null))) as average, stdev(eval(if(_time < relative_time(now(), "-1d"),count,null))) as stdev by user
- | where latest>stdev+average

# Technique: Time Series Detection

## Other Variations: # of Patient Records Viewed

- index=health_logs sourcetype=record:access
- | bin span=1d _time | stats dc(patient_id) as count by user _time
- | stats max(eval(if(_time >= relative_time(now(), "1d"), count, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d"),count,null))) as average, stdev(eval(if(_time < relative_time(now(), "-1d"),count,null))) as stdev by user
- | where latest>stdev+average

# Technique: Time Series Detection

# Technique: Time Series Detection

## Wait, isn't Standard Deviation only Accurate for Normal Distributions?

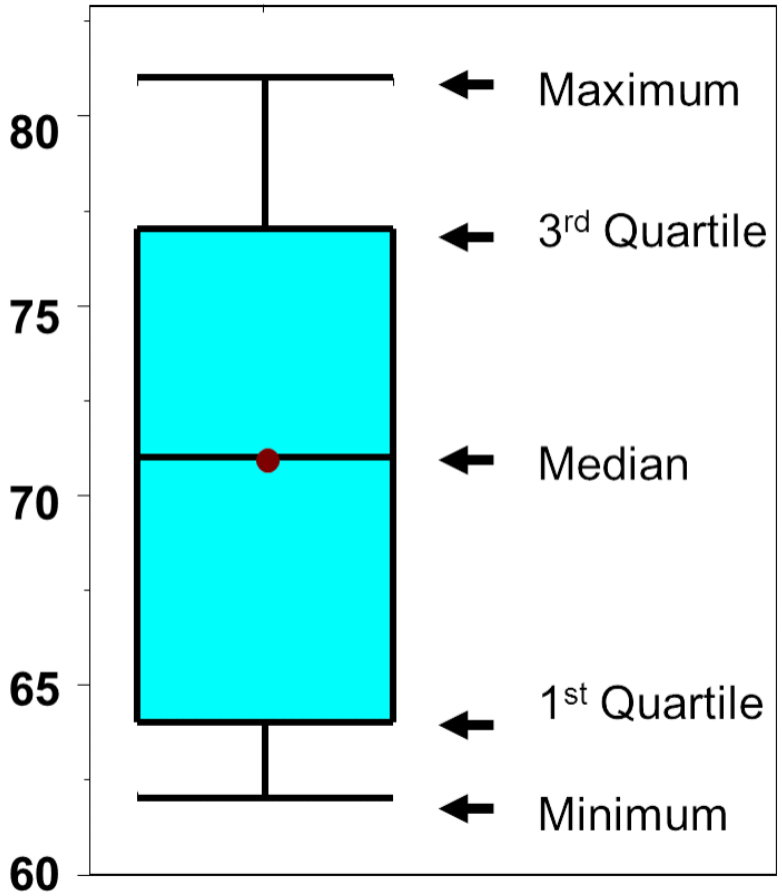
- If you have ever said sentences like the one in the header above, then this slide's for you!
- The traditional use case for Standard Deviation is predicting exactly what percentage of a population fits into a certain fraction. For example, you might set a bar at 2 standard deviations and know that means 5% of the population fits. Or 3 standard deviations, for 0.3% of the population. (Just google 3 standard deviations).
- However, this is entirely dependent on having a normal distribution. A normal distribution is that stereo-typical bell curve graph that you see on stats textbooks.
- I the security world, we virtually never see a normal distribution. We see all kinds of slanted distributions, and that gives a very justified concern about relying in math that's not quite sound!
- As a result, if you were to say "I'm going to user 3 standard deviations because I only want to see users who fit into the top 0.3%" you would definitely be misled. However, usually in security we don't care about such specific designations. We want someone who is "anomalous" or "very unusual."
- Looking broad strokes across many datasets, my general rule of thumb is that if someone is above 3 stdev, then they're "anomalous". Above 6 stdev and they're "suspicious." Anomalous things, don't send directly to the SOC, just track it. Above 6 stdev, send to the SOC.
- That rule of thumb may not fit your dataset, but it's generally pretty easy to find out - just look at your data to see what kind of events show up at each threshold. If you see 6 stdev and think, "ah, that's probably not black-and-white enough to send to the SOC" then go to 10 stdev.
- Of course, you can also take a different approach to time series detection and not use StDev, it's not the only option…

splunk> .conf2017

# Technique: Time Series Detection

## An Alternative to StDev: Inter-Quartile Range

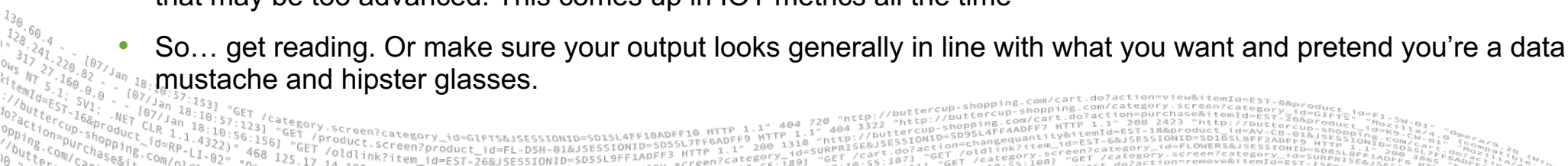
- IQR queries are a bit easier to understand conceptually, and they aren't swayed by dataset extremes. They calculate the difference between the 25th percentile and the 75th percentile, let's call it X. Then they look for any data points more than X above the 75th precentile.
- Just like with StDev, we still have a coefficient - with stdev you look for datapoints 6 stdev above the average, here you might look for items 1.5, 3, or 6 IQRs above the 75th percentile.
- In my experience, I prefer stdev because I do care about including the outliers in my variance calculation, but it's purely preference. I have asked many different people with PhDs and data science degrees, and there's never been a concrete difference.
- For an example using IQR, check out the Machine Learning Toolkit example at the end of this presentation.



# Technique: Time Series Detection

## Other Alternatives to StDev

- Sometimes IQR and StDev just aren't the right conceptual choices - for a particular dataset the data variance doesn't quite fit. Here are a couple of other techniques to keep in mind.
- Comparative Ratios: In our "Search: When Log Sources Go Quiet" example later in this doc, we don't look at the # of Windows Security Logs, we look at the ratio of Windows Security Logs to overall logs. That will provide much more accurate results.
- Normalizing data via log: https://www.r-statistics.com/2013/05/log-transformations-for-skewed-and-wide-distributions-from-practical-data-science-with-r/
- I also chatted with one of the Splunk ML experts, Andrew Stein, who told me:
  You may also consider Kolmogorov-Smirnov. Given two probability distributions (one reference, one unknown) you can measure how similar are the two. So make your reference = normal, and your unknown the observed one. For example I could measure what a specific time series is verse a normal distribution (spl for normal is on bbo.com ) using KS or whatever and I could tell if a time series was "normal"
  - He also called out: https://conf.splunk.com/files/2016/slides/a-very-brief-introduction-to-machine-learning-for-itoa.pdf
  - And http://shahramabyari.com/2015/12/21/data-preparation-for-predictive-modeling-resolving-skewness/
  - And https://en.wikipedia.org/wiki/Kolmogorov-Smirnov_test
  - And that can also smooth any time series in Splunk using http://docs.splunk.com/Documentation/MLApp/2.3.0/API/SavitzyGolayFilter but that may be too advanced. This comes up in IOT metrics all the time
  - So… get reading. Or make sure your output looks generally in line with what you want and pretend you're a data scientist. You will need a mustache and hipster glasses.



splunk> .conf2017

# Technique: Time Series Detection

## Three Last Thoughts

- Ultimately there's no magic number of stdev, or IQR, or etc. Experiment with your data and see what comes out.
- When you are using any behavioral profile, a key concern is confidence checking. After all, we don't want to overwhelm the SOC with noise.
  - Go check out the "Technique: Confidence Checking" elsewhere in this doc.
- Scaling this kind of detection is also very important for even medium sized organizations!
  - Go check out "Technique: Summary Indexing" elsewhere in this doc.

# Technique: Time Series * First Time Seen Detection

## Background and Challenges

- "Wait, first logon to new server? No. No! I don't want that running in my network! That is way too noisy! Stay away from me!"
- As detailed in the First Time Seen Detection section, we can easily detect interesting activities such as the first time a user logs into a server for the first time.
- Unfortunately, some users do this all the time. For a person in marketing who typically logs into 4 servers a day, logging into 100 new servers is terrifying! For the IT Admin runs Patch Tuesday with some epic scripts she cooked up, not remotely interesting.
- One way to approach this is to combine the above two approaches to not alert every time someone logs into a new server, but alert if someone logs into more new servers than they typically do.

# Technique: Time Series * First Time Seen Detection

## Bam! Bringing it all together

- We've already explained how both of these searches in the prior two sections - the only part we've covered less is Summary Indexing which you can take a look at in the section of the same name.

Detect first time logons

```
sourcetype=win*security
| stats earliest(_time) as earliest latest(_time) as latest
   by user, dest
| inputlookup append=t lookup_cache.csv
| stats min(earliest) as earliest max(latest) as latest
  by  user, dest
| outputlookup lookup_cache.csv
| where if(earliest>=relative_time(now(), "-1d@d"), 1, 0)

| eval alerttype="FirstTimeSeen",
alertname="FirstLogonToNewServer"

| collect index=anomalies
```

Detect an anomalous number for a user, generate a *single* alert

```
Index=anomalies alerttype="FirstTimeSeen"
alertname="FirstLogonToNewServer"
| bin span=1d _time | stats count by user _time
| stats
 max(eval(if(_time >= relative_time(now(), "1d"), count, null))) as latest
 avg(eval(if(_time < relative_time(now(), "-1d"),count,null))) as average,
 stdev(eval(if(_time < relative_time(now(), "-1d"),count,null))) as stdev
by user
| where latest>3*stdev+average
```

Another approach here would be to maintain a lookup that showed the avg number of servers any given user logs into per day. That way you can factor that into how you handle the alert

splunk> .conf2017

# NINJA Techniques

Time to put on your sunglasses

splunk> .conf2017

# Technique: tstats

## Background and Challenges

- "I want to be able to look at all the data. I mean ALL the data."
- "I'm running 5+ concurrent correlation searches at all times and need to speed them up!"
- "We are spending more on Splunk so that we can afford the correlation searches we have to run over our massive dataset."

splunk> .conf2017

# Technique: tstats

tstats speeds search dramatically.

## Raw Search: 21 Seconds

[search tag=malware earliest=-20m@m latest=-15m@m | table dest | rename dest as src ]

earliest=-20m@m (sourcetype=sysmon OR sourcetype=carbon_black eventtype=process_launch) OR (sourcetype=proxy category=uncategorized)

| stats count(eval(sourcetype="proxy")) as proxy_events count(eval(sourcetype="carbon_black" OR sourcetype="sysmon")) as endpoint_events by src

| where proxy_events > 0 AND endpoint_events > 0

## tstats Search: 2 Seconds

| tstats prestats=t summariesonly=t count(Malware_Attacks.src) as malwarehits from datamodel=Malware where Malware_Attacks.action=allowed groupby Malware_Attacks.src

| tstats prestats=t append=t summariesonly=t count(web.src) as webhits from datamodel=Web where web.http_user_agent="shockwave flash" groupby web.src

| tstats prestats=t append=t summariesonly=t count(All_Changes.dest) from datamodel=Change_Analysis where sourcetype=carbon_black OR sourcetype=sysmon groupby All_Changes.dest

| rename web.src as src Malware_Attacks.src as src All_Changes.dest as src

| stats count(Malware_Attacks.src) as malwarehits count(web.src) as webhits count(All_Changes.dest) as process_launches by src

splunk> .conf2017

# Technique: tstats

tstats speeds search dramatically.

## Raw Search: 68,476 Seconds

- index=* earliest=-24h
  | bucket _time span=1h
  | stats count by sourcetype, _time

## tstats Search: 6 Seconds

- | tstats count where index=*
  by sourcetype _time span=1h

# Technique: tstats

Let's just summarize this in one slide, shall we?

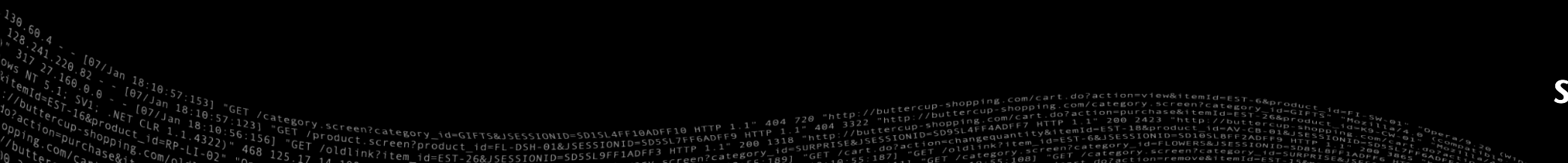
- There is far more content to cover about tstats than is reasonable in just this section, so let's go view the entire talk from conf2016, that will be repeated at conf 2017.
- Check out: http://dvsplunk.com/ninjutsu

# Technique: Timestamps and Timestamps

## Background and Challenges

- "How can I analyze data that was just ingested, regardless of timestamp?"
- "How much of my data is coming in with incorrect timestamps?"
- "How much of my data is coming in with future timestamps?"

- All data is indexed with _time, but we also add _indextime which shows the time that the data was indexed. This is powerful!



splunk> .conf2017

# Technique: Timestamps and Timestamps

## Searching Data that was just Indexed

- When you query _time, you hit earliest. When you query _indextime, you hit _indextime. These earliest/latest settings both apply, but keep the _time as low as possible because that defines what indexes you need to search through.

```
index=unstable_timestamps

    earliest=-24h
    latest=+24h

    _index_earliest=-60m

| table fields you care about
```

This is a slower approach, so only use when you have known unstable indexes

You should specify a big _time window that you know will encompass the time instability, but as little more as possible

Then include your _index_earliest

Finally, continue with your search

splunk> .conf2017

# Technique: Timestamps and Timestamps

## Searching Data with Very Old Timestamps

- You can look very old timestamps that was just indexed, though beware that this search is extremely slow because it needs to look through every old bucket you have. It's prudent to run this periodically as an all time real time search (if you have Indexed Real-time turned on) or occassionally to test.

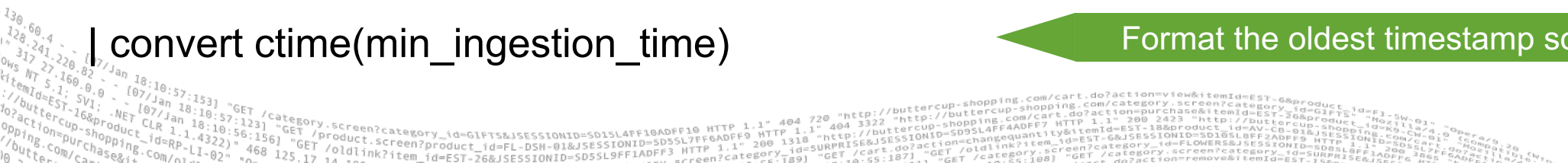
```
| tstats count min(_time) as min_ingestion_time

    where index=*

    earliest=0 latest=-1h

    _index_earliest=-5m _index_latest=now

by host sourcetype

| convert ctime(min_ingestion_time)
```

Use tstats to quickly parse out the count and the min(_time)

Pick a giant _time window. You may want to start with latest=-8h to get the low hanging fruit.

Then include your _index_earliest

Format the oldest timestamp so that it's readable



splunk> .conf2017

# Technique: Timestamps and Timestamps

## Searching Data with Future Timestamps

- Looking at events with future timestamps is typically very fruitful for finding incorrect timestamps.

```
| tstats count max(_time) as max_ingestion_time
    where index=*
        earliest=+30s latest=+20y
        _index_earliest=-5m _index_latest=now
    by host sourcetype
    | convert ctime(max_ingestion_time)
```

Use tstats to quickly parse out the count and the min(_time)

Pick systems 30 seconds

Then include your _index_earliest

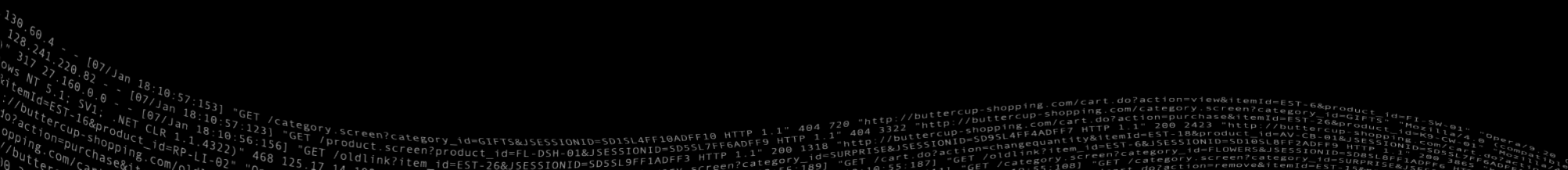
Format the oldest timestamp so that it's readable

splunk> .conf2017

# Technique: Advanced Search Commands

## Background and Challenges

- "I want to run several different sets of reporting commands from the same search"
  - E.g., "I want to update this lookup with one subset of the dataset, then run anomaly detection on a different subset"
- "I want to search multiple datasets, and apply different streaming commands to each set"
- "I don’t know what my fields will be named, but I have to manipulate them anyway!"
- "I want to do lots of other *weird* stuff"



splunk> .conf2017

# Technique: Advanced Search Commands

## The multisearch Command

- We all know you can do index=a OR index=b, but if you have to transform each differently this becomes a major hassle.
- Multisearch will actually run two different searches, but bring the results together for you.

```
| multisearch

  [ index=ips |
`lower_severity_for_low_confidence`]

  [ index=sandbox_confirmed |
`raise_severity_for_high_confidence`]

| stats max(severity) values(index) by host
```

You start with | multisearch, so that Splunk knows you don't want to start with a normal search

Just like subsearches, put each search in square brackets. You can use any streaming search commands

Any additional searches can be put in additional sets of angle brackets

Finally, continue with your search

# Technique: Advanced Search Commands

## The multireport Command

- When running multiple searches over the same dataset, use case developers have to consider "is this something that can be combined." While you don't want to go crazy (consider Multi-Scenario Alerts in this document), you can get much more convergence with multireport.
- I find this most useful when you have one search that leverages a stored lookup (or you build a lookup to provide context to an analyst), but you also want to update the stored lookup without managing another search. But there are many use cases.

```
index=proxy
| multireport

 [ | stats values(domain) count min(_time) max(_time) by user
| outputlookup contextual_per_user_info.csv | where
hide="TheseEvents"]

 [ | search category=adult | collect index=hr | where
hide="TheseEvents"]

 [ | lookup threatIntel domain | search threat_hit=*]
```

Multireport forks off multiple searches

Each search sits in a set of square brackets. We can do whatever we want to in here, including outputlookups, collects, etc.
Multireport will append the output of each search, so here we use the | where clause that will hide all results from the analyst

This final search is the one I actually want sent to the user (or correlation search, etc.), so no | where

splunk> .conf2017

# Technique: Advanced Search Commands

## The foreach Command

- The foreach command is great for two things: one is saving yourself copy-paste work to apply the same change to many fields, and the other is manipulating fields whose name you don't know.
- Foreach works by taking a list of fields (or *) and then a set of streaming commands to run. (What's a streaming command? https://docs.splunk.com/Documentation/Splunk/6.6.2/Search/Typesofcommands)
- Remember with weird field names in eval, double quotes on the left side, single quotes on the right side. So: | eval "<<FIELD>>_value" = "The value for <<FIELD>> is: " . '<<FIELD>>'

### Repeat Operations

```
Index=business_operations sourcetype=hourly_data
| stats avg(metric_*) as hourly_average_*
| foreach hourly_average_*
    [| eval "<<FIELD>>" = round('<<FIELD>>' , 2) ]
```

### Unknown Field Names

This search will track what fields are defined in every sourcetype in your environment. Pretty useful, right? Truthfully, most times this scenario comes up at the end of half a page of SPL converting some terrible XML or what have you… this is simpler.

```
index=*
| fields - _raw whatever other default fields we don't want
| eval field_names = ""
| foreach * [ eval field_names = mvappend(field_names, "<<FIELD>>")]
| stats values(field_names) by sourcetype
```

splunk> .conf2017

# Technique: Advanced Search Commands

## These are cool! What else do you have?

- One of my favorite .conf slide decks of all time is Lesser Known Search Commands by Kyle Smith, at .conf 2016
- Not only is Kyle a proud fez-wearing man, his talk also walks through really really powerful tools. I *highly* recommend it!

**Lesser Known Search Commands**

**Wednesday, September 28, 2016** | **3:30 PM-4:15 PM**

**Thursday, September 29, 2016** | **1:30 PM-2:15 PM**

INTERMEDIATE | **Products:** Splunk Enterprise | **Role:** Data Scientist/Analyst, Splunk Technical Champion, Administrator, Security Analyst | **Track:** Splunk Foundations | **Session Focus:** Search Language | **Other Topics:** Best Practices

**Speakers**

**Kyle Smith**, Integration Developer, Aplura

- Slides: http://conf.splunk.com/files/2016/slides/lesser-known-search-commands.pdf
- Recording: http://conf.splunk.com/files/2016/recordings/lesser-known-search-commands.mp4

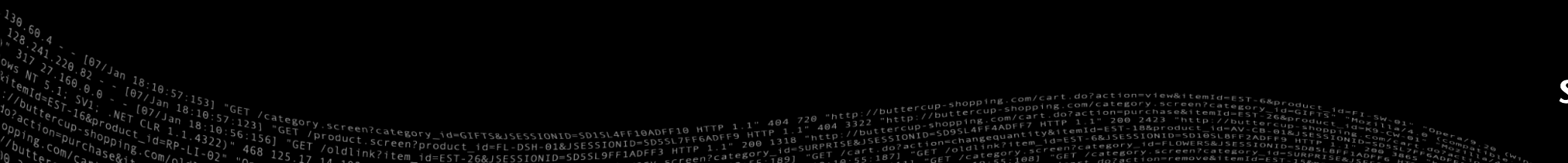
splunk> .conf2017

# Technique: Metacharacteristics

## Background and Challenges

- "I'm on a hunting expedition and I want to look for unusual {user agents, urls, etc}"
- "I know most of these events are similar, which ones are different?"

- Metacharacteristics are a concept I first heard from Monzy Merza, our Head of Security Research, and basically refer to objective numeric measures you could use to measure a string so that you can find unusual strings in a big series.
- Examples here include length, punctuation, frequency, temporality, or coherence.
- Few organizations really come to use these, but for the true Ninja who has conquered all low hanging fruit, these can be the gateway to something great.



# Technique: Metacharacteristics

## Shape

- Shape can be the length of a URL, the punct of a URL etc.
  - http://myurl.com/codepath
  - http://myurl.com/codepath?query=Robert%2527)%3b%2520DROP%2520TABLE%2520Students%3b
- Use eval with len (length), punct, and replace

# Technique: Metacharacteristics

## Frequency

- Understand your common ratios is easy - HTTP GET/POST/Connect/Delete. Track the # of GETs vs POSTs, the # of text/html vs application/octet-stream
- You've been around for 2,5,10,20 years. Track how often you talk to different websites, and alert on newness
- Detect with top/rare/stats/timechart
- Leverage the First Time Seen detection and Time Series Analysis detections from this doc for applying some of these capabilities in a correlation search.

# Technique: Metacharacteristics

## Temporality

- Long URLs typically immediately follow short urls (or are to advertising servers)
- Examples:
  - https://goo.gl/maps/yjXdP
  - https://www.google.com/maps/place/270+Brannan+St[202 characters clipped]
- Detect with: streamstats
- Many activities occur only during 9-5, 8-6, or etc.
- Detect with: date_hour (if not global) or eval's strftime()

# Technique: Metacharacteristics

## Coherence

- Coherence (in this case) - Systems that are servers tend to stay servers, systems that are clients tend to stay clients. Systems that don’t generate a lot of denies don’t tend to be denies.
- Also useful for looking at network traffic

# Technique: Metacharacteristics

## Example One

(No Explanation? Welcome to Ninja Section! Also, it's hard, and this presentation is due tomorrow!)

```
| tstats summariesonly=t count from datamodel=Network_Sessions where src!=dest
earliest=-30d@d groupby All_Sessions.src_ip All_Sessions.dest_ip _time span=1d  |
eval pairs = mvappend("src|" + 'All_Sessions.src_ip', "dest|" + 'All_Sessions.dest_ip')  |
fields pairs _time | mvexpand pairs | rex field=pairs "(?<direction>.*?)\|(?<host>.*)" |
bucket _time span=1d | stats count(eval(direction="src")) as initiating
count(eval(direction="dest")) as terminating by host _time | eval isRecent =
if(_time>relative_time(now(), "-1d"), "yes", "no") | eval ratio = initiating /
(initiating+terminating) |  stats avg(eval(if(isRecent="no", ratio, null))) as avg_ratio
avg(eval(if(isRecent="yes", ratio, null))) as recent_ratio by host | where (avg_ratio > 0.9
AND recent_ratio < 0.3) OR (avg_ratio < 0.1 AND recent_ratio > 0.7)
```

This search has completed and has returned **759** results by scanning **128,884,198** events in **52.932** seconds.

# Technique: Metacharacteristics

## Example Two

```
| tstats prestats=t summariesonly=t count(All_Sessions.src_ip) from
datamodel=Network_Sessions where All_Sessions.src_ip!=All_Sessions.dest_ip
All_Sessions.src_ip=* earliest=-30d@d groupby All_Sessions.src_ip _time span=1d |
tstats prestats=t append=t summariesonly=t count(All_Sessions.dest_ip) from
datamodel=Network_Sessions where All_Sessions.src_ip!=All_Sessions.dest_ip
All_Sessions.dest_ip=* earliest=-30d@d groupby All_Sessions.dest_ip _time span=1d |
rename All_Sessions.src_ip as ip All_Sessions.dest_ip as ip |  bucket _time span=1d |
stats count(All_Sessions.src_ip) as initiating count(All_Sessions.dest_ip) as terminating
by ip  _time | eval isRecent = if(_time>relative_time(now(), "-1d"), "yes", "no") | eval ratio
= coalesce(initiating,0) / (coalesce(initiating,0)+coalesce(terminating,0)) | where
isnotnull(ratio) |  stats sum(initiating) sum(terminating) avg(eval(if(isRecent="no", ratio,
null))) as avg_ratio avg(eval(if(isRecent="yes", ratio, null))) as recent_ratio by ip | where
isnotnull(recent_ratio) AND isnotnull(avg_ratio) | where (avg_ratio > 0.9 AND
recent_ratio < 0.8) OR (avg_ratio < 0.1 AND recent_ratio > 0.2)
```

# Technique: Machine Learning Toolkit Numeric Clustering

## Background and Challenges

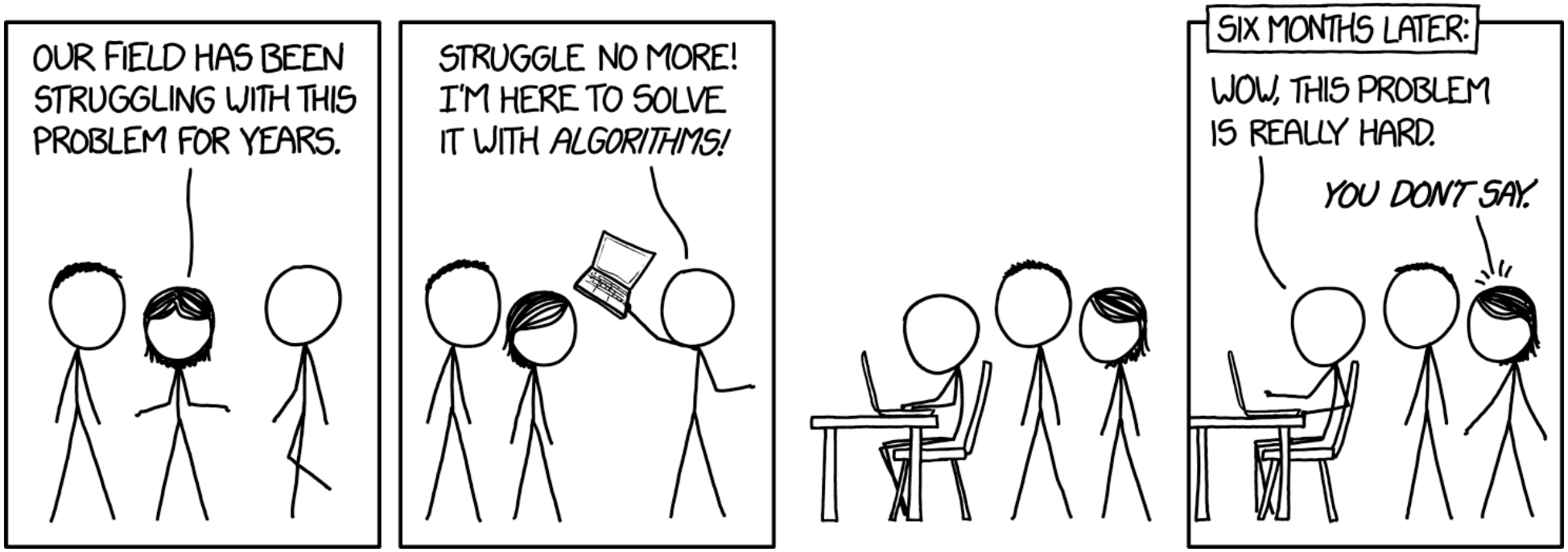
- "I have a bunch of numeric measures and I want to find outliers!"
- "I am looking for new and relatively unproven ways to hunt!"
- Let's be honest: "How do I use this Machine Learning Toolkit?"

- Splunk's Machine Learning Toolkit (hereafter referred to as just MLTK) is a great way to build your own Machine Learning (ML) use cases with algorithms that are already written, and packaged in Splunk.
- The benefit is obvious - ML can allow you to do some detections you simply can't do otherwise. The downside is that there's a lot more hand waving and magic and uncertainty involved with ML than with normal Splunk.
- I highly recommend reviewing the Time Series Analysis and First Time Seen analysis sections of this doc before diving into this.

splunk> .conf2017

# Technique: Machine Learning Toolkit Numeric Clustering

## Obligatory xkcd #1




https://xkcd.com/1831/

- When approaching difficult problems with Machine Learning, remember that they are difficult problems for a reason. ML isn't a magic wand, ML doesn't fix problems that you don't understand. ML's greatest skill is to take an understood solution that would be impossible with your manpower or existing computation and then scale it higher.

# Technique: Machine Learning Toolkit Numeric Clustering

## Best Known Examples of Security MLTK

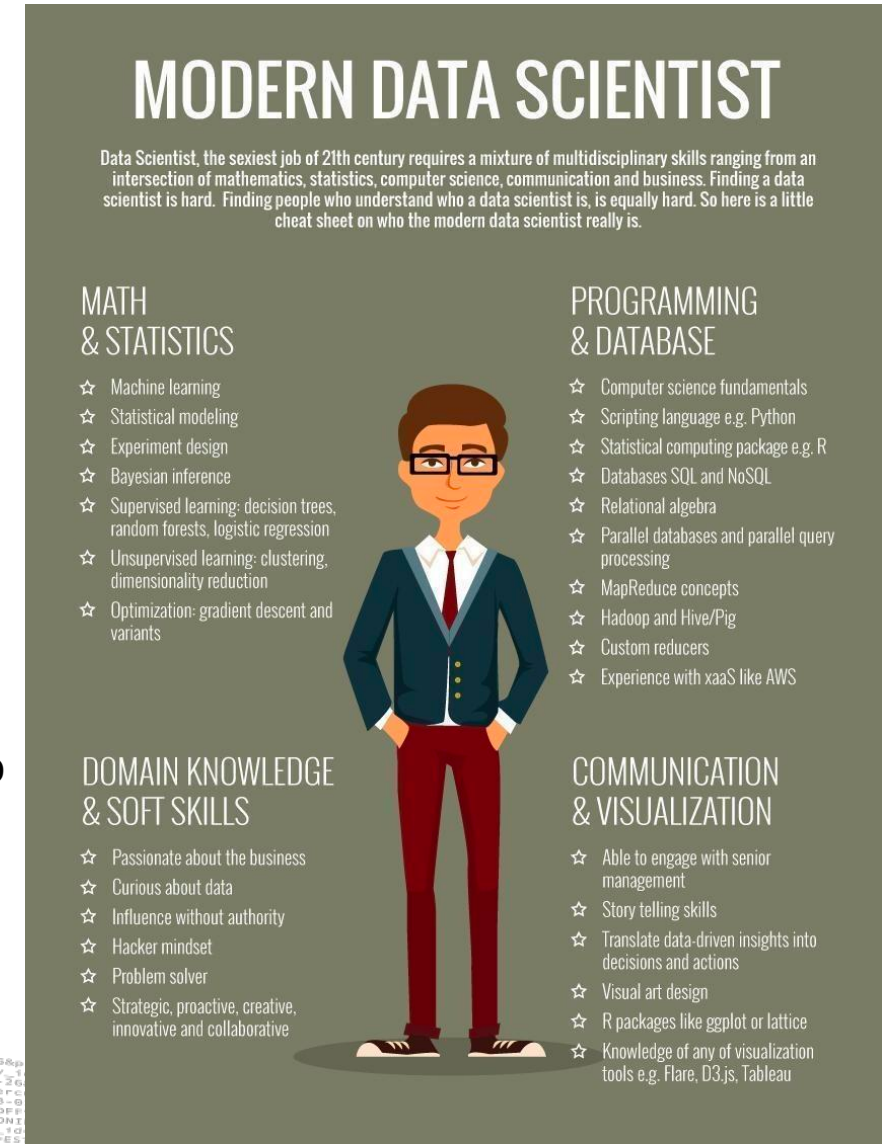
- Security MLTK adoption has been slower than general MTLK adoption, but we have two examples of using MLTK that seem very solid: supervised detection of malicious domains built by Philipp Drieger out of Munich, and unsupervised clustering of numeric time series data built jointly by US Splunk Security and ML Specialists.
- Philipp's Malicious Domain detection is the best example of demonstrating how to use the MLTK that I have ever seen. The core scenario is that we can use domains that are known to be a part of botnets to predict the qualities of future C2 domains for those botnets. Philipp found an open source list of 50k domains with the associated families, and then converted those to "features" (next slide) using eval, URL Toolbox, and also MLTK. Then he tried several clustering algorithms to see which gave him the best accuracy. Once he had the model, he tried it out on a much larger list to track the true/false positive/negative ratios. This is great work that could be used by advanced organizations
  - … waiting on link to content … … you can always ask your Splunk team to show it to you!
- The Security + ML Specialist teams worked together to identify anomalies in Salesforce.com (SFDC) audit data (which is a decent proxy for any three tier application server). The idea was to detect users who were acting unusually in SFDC, specifically with the indication that they were going to exfiltrate data. This is the scenario that we will go through over the following slides.

splunk> .conf2017

# Technique: Machine Learning Toolkit Numeric Clustering

## Machine Learning Vocab - Features and and Supervised

- There is a slightly different vocabulary when we talk Machine Learning - both IT/Security and Machine Learning are practices that have been around for many decades, but they grew up independently and so developed their own dialects. So that you can follow Machine Learning discussions, here are the key terms:
- Entity - whatever it is you're analyzing. Might be a user, might be a computer, might be a virus signature.
- Features - these are the fields that you will analyze in your input data.
- Feature Selection - this is the process of deciding what fields to analyze. For example, I checked a few days of SFDC logs and found 1164 fields - how do you decide what will be impactful?
- Supervised - an algorithm where you are labeling your input data as being "good/bad" or by malware family, etc. Over-simplified version: "Do you have examples of what you want to detect? Supervised."
- Unsupervised - an algorithm where you don't label your input data, and the algorithm itself just comes up with answers. Over-simplified version: "You don't have examples of what you want to detect? Unsupervised."



# Technique: Machine Learning Toolkit Numeric Clustering

## SFDC Data Example

- SFDC has an Event Log File (additional cost) audit log that allows you to see what your users are doing inside of Salesforce. It creates a daily dump listing activities by user. So when you run a query, you can see each request made, each API connection

- 2017-02-06T15:52:09.200+0000 SFDCLogType="DocumentAttachmentDownloads" SFDCLogId="0AT33000000UCXqGAK" SFDCLogDate="2017-02-06T00:00:00.000+0000" TIMESTAMP_DERIVED="2017-02-06T15:52:09.200Z" REQUEST_ID="491x_Vi5XxDS1AVeRhRaXF" EVENT_TYPE="DocumentAttachmentDownloads" USER_ID_DERIVED="005400000083CQSAA2" USER_ID="005400000083CQS" ENTITY_ID="01540000000Mc72" FILE_NAME="Splunk - CRM.png" FILE_TYPE="image/png" TIMESTAMP="20170206155209.200" ORGANIZATION_ID="00D400000003VqL"

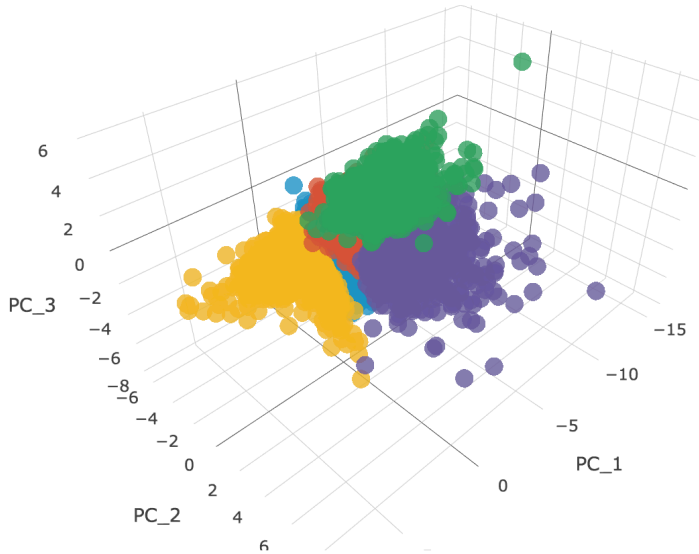
- 2017-02-06T15:52:08.374+0000 SFDCLogType="URI" SFDCLogId="0AT33000000MhY5GQK" SFDCLogDate="2017-02-06T00:00:00.000+0000" DB_BLOCKS="16113" REQUEST_STATUS="S" RUN_TIME="989" USER_ID_DERIVED="005400000083CQSAA2" REFERRER_URI="[...]" URI="/home/home.jsp" URI_ID_DERIVED="" DB_TOTAL_TIME="582637378" USER_ID="005400000083CQS" SESSION_KEY="G/lti8OvlcoIBd1R" CLIENT_IP="[...]" REQUEST_ID="491x_P33DCVcFfVeRhqoSk" DB_CPU_TIME="420" EVENT_TYPE="URI" LOGIN_KEY="aGsG3ecCQGKmHgdL" TIMESTAMP_DERIVED="2017-02-06T15:52:08.374Z" ORGANIZATION_ID="00D400000003VqL" TIMESTAMP="20170206155208.374" CPU_TIME="344"

- 2017-02-06T15:52:05.547+0000 SFDCLogType="Login" SFDCLogId="0AT33000000UhXuGAK" SFDCLogDate="2017-02-06T00:00:00.000+0000" BROWSER_TYPE="Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.95 Safari/537.36" REQUEST_STATUS="" USER_ID_DERIVED="005400000083CQSAA2" URI="/index.jsp" DB_TOTAL_TIME="76331254" CLIENT_IP="[myip]" LOGIN_KEY="" SOURCE_IP="[myip]" API_TYPE="" CPU_TIME="44" SESSION_KEY="" RUN_TIME="133" CIPHER_SUITE="ECDHE-RSA-AES256-GCM-SHA384" USER_ID="005400000083CQS" TIMESTAMP_DERIVED="2017-02-06T15:52:05.547Z" API_VERSION="9998.0" REQUEST_ID="491x_Hi2VageXqMf12zgiXpF" USER_NAME="david.veuve" EVENT_TYPE="Login" URI_ID_DERIVED="" TLS_PROTOCOL="TLSv1.2" TIMESTAMP="20170206155205.547" ORGANIZATION_ID="00D400000003VqL"

# Technique: Machine Learning Toolkit Numeric Clustering

## Defining the Overall Plan

- The first thing you typically need to do with ML is feature selection. Inside of Splunk, we typically do this via stats. I just picked out events that looked interesting.
- Then we need to normalize the data. In this case, we want to normalize on a per-user basis, so that we maintain the variation for a single user but overall keep visibility when a user suddenly changes dramatically. The core requirement here is to be able to detect a user who rarely uses SFDC and then starts exporting key contacts, while also detecting a power user who exports all of the contacts and all of the opportunities and etc. There are a few ways to do this, but we will use eventstats to normalize to stdev on a per-user basis.
- Invariably then, we will want to simplify the number of fields that we are analyzing. Each additional field adds a lot of compute time for the down-stream analysis. The most common way to do that is with an algorithm called PCA (Principal Component Analysis). PCA will take multiple fields (in this case 11) and compress them down into X number of fields (in this case 5). This is a lossy compression, so we do lose some detail in the variability, but it tries to capture how much the fields move. You can think of this as compressing a music file - you can use FLAC and it will take more time and more space but lose no quality, you can use a 256 bit MP3 and you will lose little quality but compress well, or you can use 32 bit MP3 and takes little space but sounds awful. We want to shoot for 256 bit MP3.
- Now we're ready to get really data-sciency. It's time to use k-means to cluster our data together. What we will end up with is a few very large clusters with most of our data points - we can think of these clusters as "people acting normally." Then we will run into a few nodes that are technically a part of that cluster but very very far from the cluster center - those are interesting. Also interesting are very small clusters (only a few data points), as they're by definition anomalous. Notably, if you talk to data scientists (if you *are* a data scientist!) one of the most common questions about this approach is "how do you decide on your k." K-means groups things into k clusters - you have to tell it how many clusters to make. We jointly decided on 5 for this use case, with the option to further tune. For more, see "Downsides to Building it Yourself" at the end of this section.
- To figure out which cluster points we actually want to return, we will use Inter-Quartile Range. IQR (detailed in the Time Series Analysis section) will help us determine the amount of variation in a typical cluster, and find the outliers (and by how much they're outliers). It is not heavily swayed by a long tail, so a few very distant outliers won't affect how it views the core group. See that screenshot of a 3D scatterplot of our dataset? We want the Green distant dot, and probably some of those Purple ones too.
- Finally we will use | where to look for either small clusters, or nodes that are very far from their cluster. For the former, we will have a static threshold for how many members in a cluster counts as "small" and for the latter we will use a coefficient that we can tune based on the results we are seeing.




splunk> .conf2017

# Technique: Machine Learning Toolkit Numeric Clustering

## Building the Base Dataset

- Depending on who you ask, 50-90% of a Data Scientist's time is spent collecting, ETLing and formatting data. Splunk makes that exceedingly easy.
- Below I parsed through SFDC data to pull out the individual fields I felt most likely return Security Value. I then ran an | outputlookup so that I could feed it to the ML algorithm repeatedly.

```
index=sfdc
| bucket _time span=1d
| stats dc(eval(if(like(URI_ID_DERIVED, "00140000%"), URI_ID_DERIVED, null))) as NumAccounts
        dc(eval(if(like(URI_ID_DERIVED, "0063300%"), URI_ID_DERIVED, null))) as NumOpts
        sum(ROWS_PROCESSED) as ROWS_PROCESSED
        count(eval(EVENT_TYPE="Login")) as Logins
        count(eval(EVENT_TYPE="Report")) as ReportsIssued
        count(eval(EVENT_TYPE="API" OR EVENT_TYPE="BulkApi" OR EVENT_TYPE="RestApi")) as APICalls
        sum(DB_CPU_TIME) as DB_CPU_Time
        sum(RUN_TIME) as RUN_TIME
        sum(DB_BLOCKS) as db_blocks
        dc(CLIENT_IP) as UniqueIPs
        dc(ORGANIZATION_ID) as NumOrganizations
        dc(ENTRY_POINT) as ApexExecution_Entry_Type
by USER_ID _time
| outputlookup sfdc_aggregated_data.csv
```

splunk> .conf2017

# Technique: Machine Learning Toolkit Numeric Clustering

## Running the Detection

- Now we actually do our ML!

```
| inputlookup sfdc_aggregated_data.csv

| eventstats avg(*) as AVG_* stdev(*) as STDEV_* by USER_ID
| foreach * [ eval "Z_<<FIELD>>" = ('<<FIELD>>' - 'AVG_<<FIELD>>' )
/ 'STDEV_<<FIELD>>'] | fields - AVG_* STDEV_*  | fillnull

| fit PCA k=5 Z_*

| fit KMeans k=5 PC_*

| eventstats max(clusterDist) as maxdistance p25(clusterDist) as
p25_clusterDist p50(clusterDist) as p50_clusterDist p75(clusterDist) as
p75_clusterDist dc(USER_ID) as NumIDs count as NumEntries by
cluster
| eval MaxDistance_For_IQR= (p75_clusterDist +
                    12 * (p75_clusterDist - p25_clusterDist))

| where NumEntries < 5 OR clusterDist > MaxDistance_For_IQR
```

We start with the lookup just created

Then we use eventstats and foreach to convert every numeric field to a Z score (how many stdev away from avg it is), normalizing per user

PCA lets us reduce from 11 fields to 5 fields

K-means clusters the PCA output

Then we use eventstats again to determine the Inter-Quartile Range of our data points versus the clusters that k-means just found.

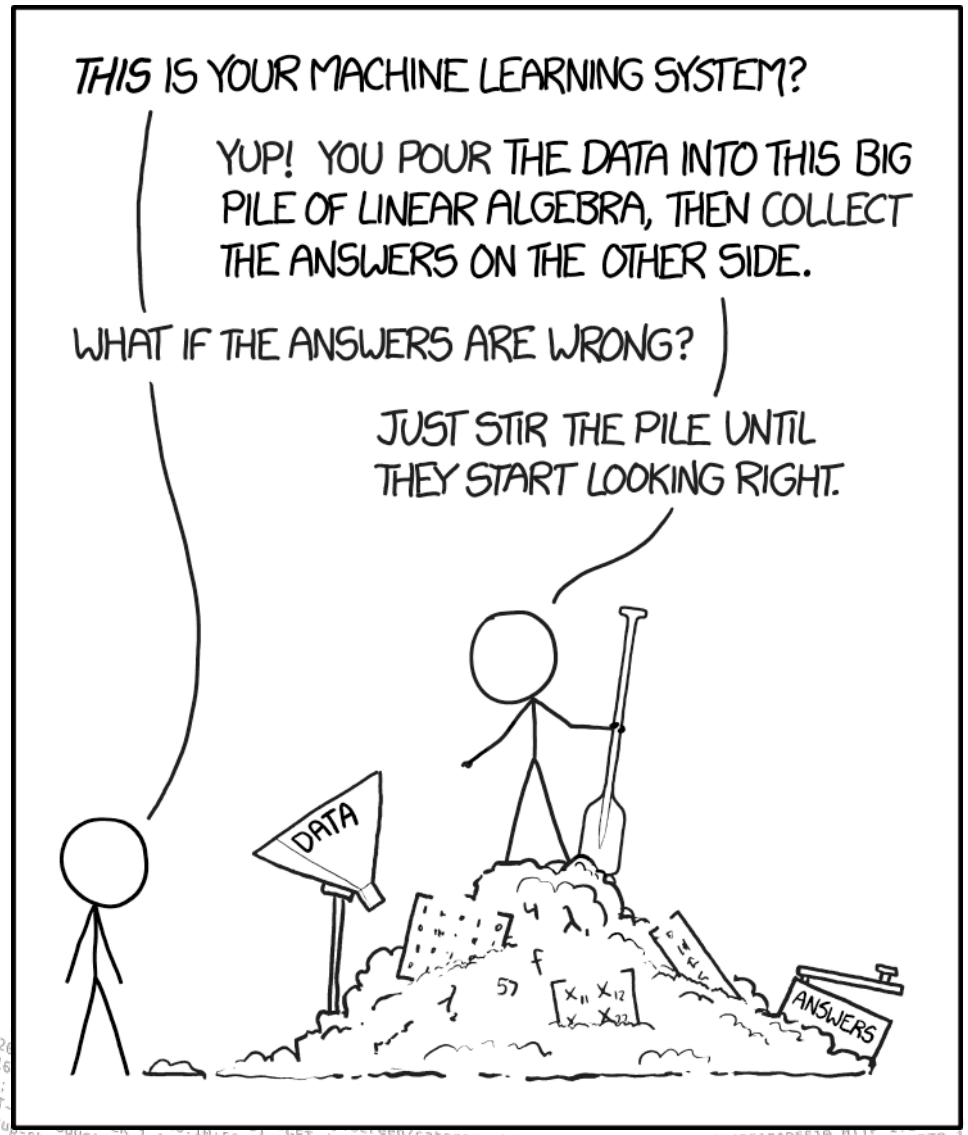
Notably, even with IQR you decide on some noise filter. Here we use 12 IQRs, a common base is 1.5.

Finally, filter for the results we want to see.

splunk> .conf2017

# Technique: Machine Learning Toolkit Numeric Clustering

## Obligatory xkcd #2




- Remember to beware of relying on any analysis that you don't understand. That doesn't mean that you shouldn't rely on it, but not blindly. Two ways in which this applies.
- If you are not a PhD with a solid understanding of how Machine Learning actually works, you should consult one before building your primary detection mechanisms on ML (I personally like to augment with ML, rather than rely on ML).
- If you have one working example that you feel very comfortable with, and are then going to apply it to another, make sure that second use case really resembles the first. Faulty underlying assumptions will doom any project.
- Tangentially: these rules also apply to data scientists. For example, Deep Learning is a new hot trend which is even deeper linear algebra that is far more difficult to detect. You may get ML intuitively, but you could still be just playing guesswork for Deep Learning. Know your limits.

https://xkcd.com/1831/

splunk> .conf2017

# Technique: Machine Learning Toolkit Numeric Clustering

## To Hunt or To Alert

- When I describe this use case to people, I usually tell them that it's more appropriate for hunting than for alerting, at this point in time.
- The primary reason is that this is a relatively untested scenario. We built this in the lab, and we've seen some value in realistic datasets, but we don't know how it will work in the field over hundreds of customers, like we do with the Time Series or First Time Seen analysis.
- That said, there is a big secondary reason: this event is harder for analysts to understand. This is an endemic problem in ML detections (something we work very hard to overcome in UBA). Consider the data at the bottom - this is for a user / day who was most anomalous out of 140k users/days. If you were a SOC analyst, what would you do with this alert?
- With all new categories of detection, they will usually start by being viewed by Use Case Dev / CERT / Hunt Teams, and then progress to Tier 3, Tier 2, before eventually Tier 1. I would recommend being conservative with this particular technique for now.

| APICalls | ApexExecution_Entry_Type | DB_CPU_Time | Division | Logins | NumAccounts | NumOpts | NumOrganizations | ROWS_PROCESSED |
|---|---|---|---|---|---|---|---|---|
| 1502 | 236 | 141830 | Sales | 12 | 11 | 66 | 1 | 449680 |

| RUN_TIME | ReportsIssued | USER_ID | UniqueIPs | _time | db_blocks | Z_APICalls | Z_ApexExecution_Entry_Type |
|---|---|---|---|---|---|---|---|
| 1548337 | 0 | ANON_USER_2642 | 10 | 2017-01-31 | 6085184 | 6.15655608 | 6.1297244 |

| Z_DB_CPU_Time | Z_Logins | Z_NumAccounts | Z_NumOpts | Z_ROWS_PROCESSED | Z_RUN_TIME | Z_ReportsIssued | Z_UniqueIPs | Z_db_blocks |
|---|---|---|---|---|---|---|---|---|
| 6.072020867 | 5.06025 | 5.44305 | 6.0859043 | 6.2460531327 | 6.0785592733 | 0.0 | 1.733256 | 5.99768515542 |

| cluster | cluster_distance | NumEntries | NumIDs | maxdistance | perc25_cluster_distance | perc50_cluster_distance | perc75_cluster_distance | MaxClusterDistance_For_IQR |
|---|---|---|---|---|---|---|---|---|
| 3 | 161.70450241 | 3799 | 2442 | 161.704502410000 | 3.470000000000 | 5.900000000000 | 10.800000000000 | 98.76000000000 |

# Technique: Machine Learning Toolkit Numeric Clustering

## Downsides to Building It Yourself

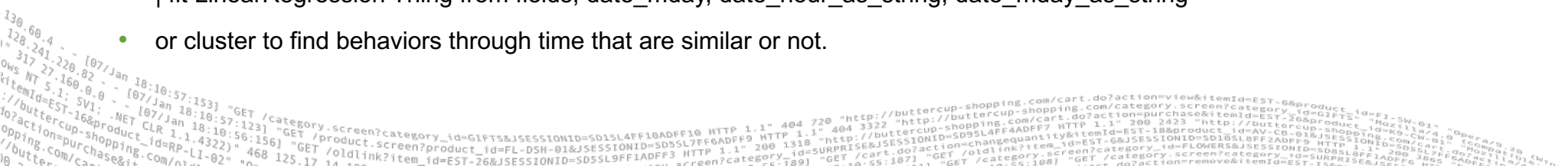
- While we did work with a data scientist to build out this model, there are *many* untested assumptions here:
  - That we are including the fields that will get us what we really care about
  - That eventstats + stdev is the right way to build a per-entity baseline
  - That 5 fields for PCA is the right number for our data source
  - That k-means is the right clustering mechanism (between core and MLTK Splunk supports four of them!)
  - That 5 clusters is the right number of clusters
  - That no scale limits (e.g., max 100k) results are being seen by MLTK
  - That 12 IQRs are the "right" number
- Ultimately there is no right or wrong answer to most of these things, but there are "more right" or "more wrong" answers. By leveraging resources available, we were able to come up with something that seems reasonable, but ultimately it depends on your data and whether you're getting valuable results. Much like with the discussion in Time Series Analysis, with Security we are not trying to get extreme precision; we're trying to get in the general ballpark so that we can then focus on the things we most care about. That means that some room for error is absolutely expected.
- However, when building out very generalizable scenarios, we recommend leveraging the work of actual data scientists. In the Splunk universe, that means Splunk UBA. You can then focus your efforts on building out use cases that aren't universal to everyone.

splunk> .conf2017

# Technique: Machine Learning Toolkit Numeric Clustering

## Other Ideas around MLTK from our ML Experts

- For time series data you often want to build a baseline over some period of time and have that baseline update (scheduled search)
  - | stats avg() as AVG stdev() as STDEV var() as VAR etc by date_hour,date_mday,ID |outputlookup mybaseline.csv
  - (advanced) maintain list of holidays as  date_mday,isHoliday
  - (advanced) remove outliers
  - (advanced) repeat with streamstats as needed (Span, global=f, current=f etc) , like https://wiki.splunk.com/Community:Plotting_a_linear_trendline with streamstats window=100 and you get a rolling R^2 between two time series. Wheeee.
  - (advanced) | calculate kurtosis, skewness, etc to represent the shape of the distribution
- in another search that you need to have run in a short time (ie not looking over a long window of time)
  - | lookup mybaseline.csv  date_hour as date_hour date_mday as date_mday ID as ID
  - | lookup holidays.csv date_mday as date_mday
  - Score the events in your new short time window with the knowledge gained from the past.
- Time series from MLTK
  - |fit linearregression | eval comment="Use the time fields in splunk as features!" | fields - comment
    | eval date_mday_as_string= date_mday."_" , comment = "date_* are numbers, but we want them treated as categorical strings" | fields - comment
    | eval date_hour_as_string = date_hour."_" , comment = "date_* are numbers, but we want them treated as categorical strings" | fields - comment
    | fit LinearRegression Thing from fields, date_mday, date_hour_as_string, date_mday_as_string
  - or cluster to find behaviors through time that are similar or not.

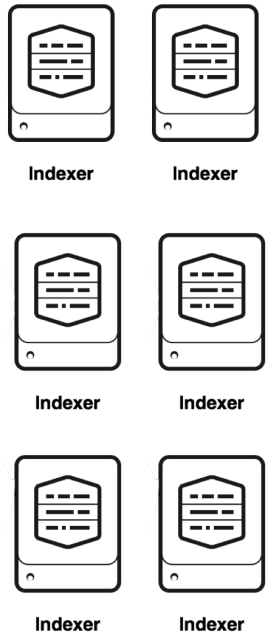


splunk> .conf2017

# Technique: Machine Learning Toolkit Numeric Clustering

## Data Reduction

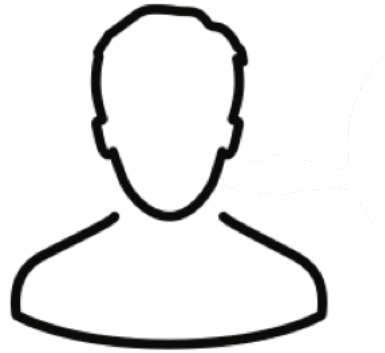
238,000,000 Events on Disk




141,000 Per User Per Day Rows

Search Head

262 Anomalous Per User Per Day Rows



Person

2.7 million events per day -> 3 anomalies per day

Need more or less? Adjust the number of IQRs

splunk> .conf2017

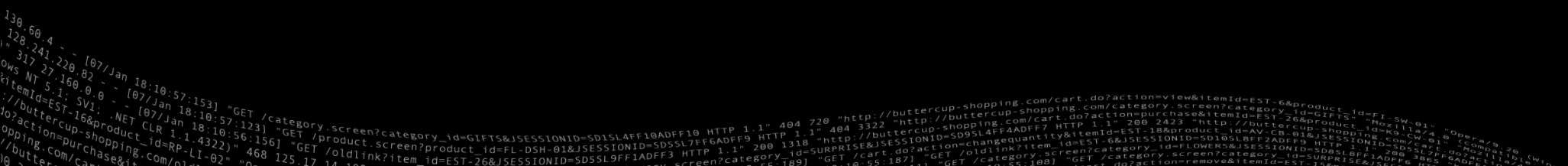
# Technique: Machine Learning Toolkit Numeric Clustering

## Now You're Trained

# Approach to Analytics

## What? Not a Technique?

- As Use Case writers get more expert, they tend to move further away from very simple use cases into more advanced anomaly detection. As we progress from "EventCode=1102" to "dc(servers) > 3 stdev + avg" value is harder to find.
- There are many conceptual approaches to dealing with this, but they all tend to boil down to the core idea of a two phase approach.
  - The first phase is to find anomalous activites, which may be good or bad. These are generally low confidence, and shouldn’t be sent to the SOC directly.
  - The second phase is to aggregate anomalies into something the SOC should view. Call them threats, or multi-vector alerts, or whatever you want - I call them threats.
- This section lays out how I view and group anomalies vs threats.



# Technical Components of Security Analytics

## Alert Aggregation

**Threat Detection**

- Manage High Volume
- Track Entity Relationships
- Combination ML + Rules

## Alert Creation

**Simpler Detection**

- Rules & Statistics
- Quick development
- Easy for analysts

**ML Based Detection**

- Detect unknown
- New vectors
- Heavy data science

## Investigation

**Investigative Platform**

- Analyst Flexibility
- Provide access to data analysis solutions
- Record historical context for everything

splunk> .conf2017

# Approach to Analytics

## Investigative Tier

Virtually every modern security detection requires some investigation, and always has. As attackers become more advanced, detection mechanisms become more advanced, it is critical to advance the investigative platform to keep pace with the new needs. The needs for investigation range from ticketing, workflow, large scale log search, the capability to ingest all of the data that will later be needed by an investigation, and more. This is the most mature of the range of capabilities required for Security Analytics success, and most organizations will have a decent investigative capability available.

While most organizations do have some basis for investigation, technology leaders must note where the key requirements differ for Security Analytics. Detections powered heavily by machine learning by definition produce more abstract results that junior level analysts have a harder time understanding. Part of the onus for auctioning these events lies with the detection logic itself providing as much context as possible to enable action, but additionally the investigative tier must be more robust to allow analysts to quickly understand a detection. This includes both the presentation of contextual information from the detection logic itself (baseline information, degree of deviation, etc.), but also a capability to more quickly explore greater amounts of data, and to have potentially relevant information surfaced.

Newer innovations to support these needs include simpler access to information (faster and more usable dashboards, form search, natural language processing), adaptive response capabilities to automate many of the menial tasks (such as acquiring forensic details, and automating remediation for predictable categories of events).

# Approach to Analytics

## Anomaly Detection Tier: Alert Creation - Simpler Detections

Everyone's first effort in the world of UEBA is to leverage rules or statistical detections. Basic approaches here are to alert if someone prints more than 100 pages, or emails more than ten documents. As organizations mature, they begin leveraging per user (or per system, per entity) baselines. This allows them to track if a user who normally prints only a few pages suddenly starts printing 75 pages - that can be an anomaly for that user, but the person who prints 200 pages a day won't be flagged unless they go far outside their normal baseline.

These detections are beneficial because they are specific to known threat vectors, and can be quickly created to detect future events within the SOC. Just as important, they are simple for security analysts to understand and action.

Within the Splunk Portfolio, the best place for simpler detections is Splunk Enterprise. Splunk customers have been using these detections for a decade, and they can be built quickly and easily. Splunk has recently doubled down on this effort and released the free Splunk Security Essentials app which delivers 50+ use cases commonly found in UEBA products. It is easy for SOC engineers to build out their own use cases leveraging time series analysis, first time seen detections, and even other advanced analytics like entropy detection, levenshtein lookalike detection, and more.

# Approach to Analytics

## Anomaly Detection Tier: Alert Creation - ML Based Detections

Many organizations have tried and failed in the past to deliver Security Analytics solutions with the simpler detections alone - while many quick wins can be attained with these technologies, they ultimately require an extreme operational expense due to the technology needs, in addition to an inability to detect novel methods. To compensate for a comparatively limited scope in detecting anomalies, teams end up doing extensive hunting, or building many rules via professional services to accommodate for scenarios that might be relevant in the future.

With Machine Learning, you can start detecting tools, techniques, or procedures that you didn't necessarily know how to predict. You can build out far more advanced technique techniques that simply aren't possible with more basic data analysis platforms. With PhD driven data science, the magnitude of detection is substantially greater.

Importantly, the recent availability of scalable ML detection doesn't reduce the need for simpler detections, the two complement each other. The simpler detections tend to produce higher confidence detections more easily actioned by SOC members for known techniques, where the advanced machine learning models can provide a backstop to approach detection from a different perspective, finding attackers the simpler detections didn't know to look for.

In the Splunk Security Portfolio, Splunk UBA is a data science platform that can facilitate these advanced anomaly detection models. Both with advanced known attack detections such as the HTTP model, that tracks known techniques in a way not possible on a lesser data science platform, or the advanced rarity and markovian models that can detect threats you didn't know how to build, Splunk UBA provides the horsepower needed to detect the suspicious actions within your environment.

# Approach to Analytics

## Anomaly Detection Tier: Alert Aggregation

The third major component of successful Security Analytics programs is an advanced threat detection capability, augmented with Machine Learning. A pre-requisite for this component is that a customer must have a successful capability for simpler rules / statistical detections, and for advanced machine learning detections. Effectively, there must be something for the threat detection to review.

Once an organization makes the investment in those initial two components, they will need to analyze a large volume of anomalous activities. It is inherent in anomaly detection technologies that there will be a great amount of noise. If that is tuned down, critical events will be missed. The solution to this is to have a second level of rules and machine learning that sits on top of the anomalies to aggregate useful events into threats. Many legacy products in this space have deployed simple rule based logic, or surfaced the users with the greatest number of threats, but these naïve approaches are insufficient.

For Splunk's Security Portfolio, Splunk UBA runs a set of machine learning powered threat models over the collection of anomalies, to surface the threats that need to be reviewed by the SOC. Even that alone is not enough - to do threat detection successfully, you also need to understand the relationships between every entity in the environment. This graph mining is a key conceptual advantage of Splunk UBA's threat models over what can be done by Splunk users directly.

# Splunk Security Portfolio

## Enterprise Security Response

- OOB key security metrics
- Incident response workflow
- Adaptive response

## Splunk Enterprise Detection

**Realm of Known**

- Log Aggregation
- Splunk Security Essentials
- Rules, statistics, correlation

## Splunk UBA Detection

**Realm of Unknown**

- Risky behavior detection
- Entity profiling, scoring
- Kill chain, graph analysis

*Human-driven*

*ML-driven*



splunk> .conf2017

# End to End Searches

## Welcome to the Full Picture

splunk> .conf2017

# Search: When Log Sources Go Quiet

## Background and Challenges

- "I need to know if my XYZ logs stop coming in"
- "ABC Malware shuts off our AV product and Windows Updates, how do I detect that?"

- Splunk users routinely need to monitor for log sources being shut off. This is typically done at both the global level (do we have latency on PAN logs) and also on a per host basis (are we getting Windows Logon events but no EDR events).

splunk> .conf2017

# Search: When Log Sources Go Quiet

## Detecting Individual Sources that Go Quiet

- Here we use a lot of tstats and stats to detect if just the Windows Security log goes offline for a host.

```
| tstats prestats=t count(host) where index=* groupby host _time
                        span=1d
| tstats prestats=t append=t count where index=*
                        sourcetype=win*security by host  _time span=1d
| stats count(host) as all_logs count as win_logs by host _time
| eval win_perc=round(100*(win_logs / all_logs), 2)
| stats count as num_data_samples
        avg(eval(if(_time<relative_time(maxtime, "-1d@d"), win_perc,
            null))) as avg
        sum(eval(if(_time<relative_time(maxtime, "-1d@d") AND
            win_perc=0, 1, null))) as past_instances_of_no_logs
        max(eval(if(_time>=relative_time(maxtime, "-1d@d"), win_perc,
            null))) as latest
    by host
| where isnotnull(avg) AND num_data_samples>10 AND
isnull(past_instances_of_no_logs) AND latest=0
```

We use a couple of tstats tricks to pull in the number of log files in general for a host, and the number of Windows Security logs

Eval calculates the percentage of Windows Security

Stats allows us to track the baseline per host

Finally, where allows us to look for new instances of no Win Security logs

splunk> .conf2017

# Search: When Log Sources Go Quiet

## Detecting Hosts That Go Quiet

- Here we look across the board for a particular host, and compare the typical time gap between periods of logs. Then we alert for excessive gaps.

```
| tstats count where index=* by host _time span=4h
```

Here tstats is giving us the number of events quickly, grouped by four hour chunks, ordered by host

```
| streamstats window=2 range(_time) as timediff by host
```

Streamstats will pull the diff

```
| stats count max(timediff) as max_timediff avg(timediff) as avg_timediff
      stdev(timediff) as stdev_timediff max(_time) as latest by host
```

Then we use stats to pull the average, stdev, and number of data samples.

```
| eval currentlag = now() - latest
```

Then we calculate the current lag

```
| where currentlag > avg_timediff*2 + stdev_timediff*6 AND count>12
```

Then we filter for hidden events.

```
| eval currentlag_in_hours=round(currentlag/3600,2)
```

Finally, let's format this stuff.

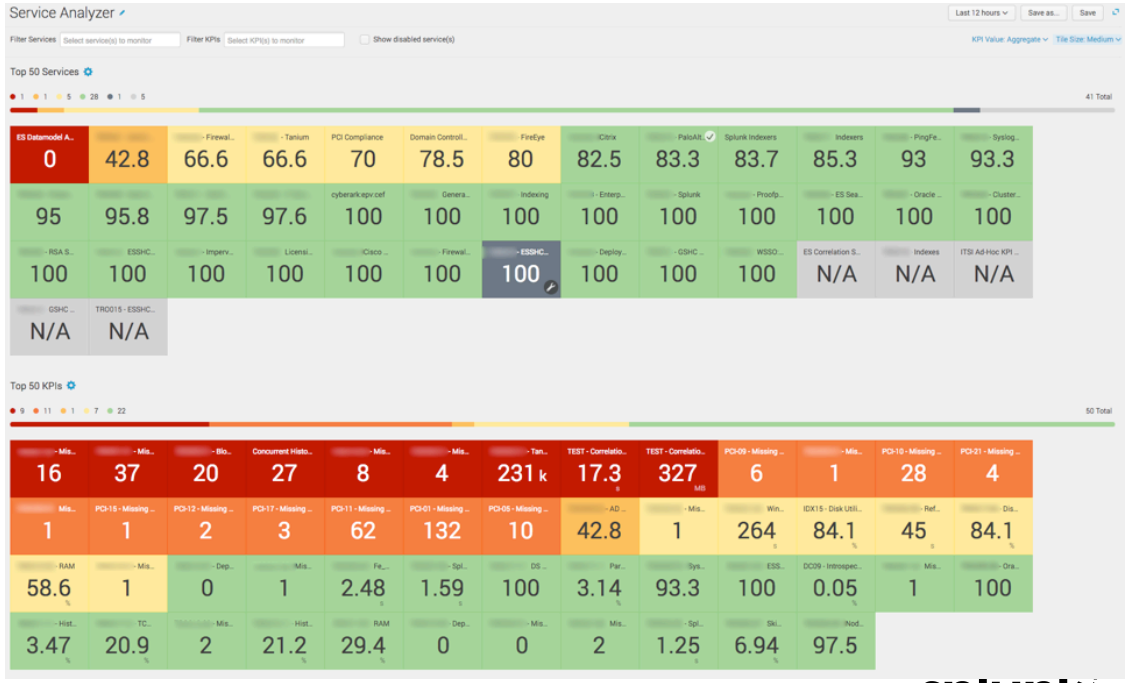
splunk> .conf2017

# Search: When Log Sources Go Quiet

## Broader Dashboard Support

- Many SOC Customers I work with have a dashboard that SOC analysts can go to to get the status of the different log sources. The goal is to let everyone know if all of a sudden Palo Alto Networks logs are delayed.
- These dashboards typically have a series of boxes with Green / Yellow Red indicators for each data source.
- Many customers have even begun using ITSI to track their data source pipelines.

  Here's a screenshot of a POC at one customer.

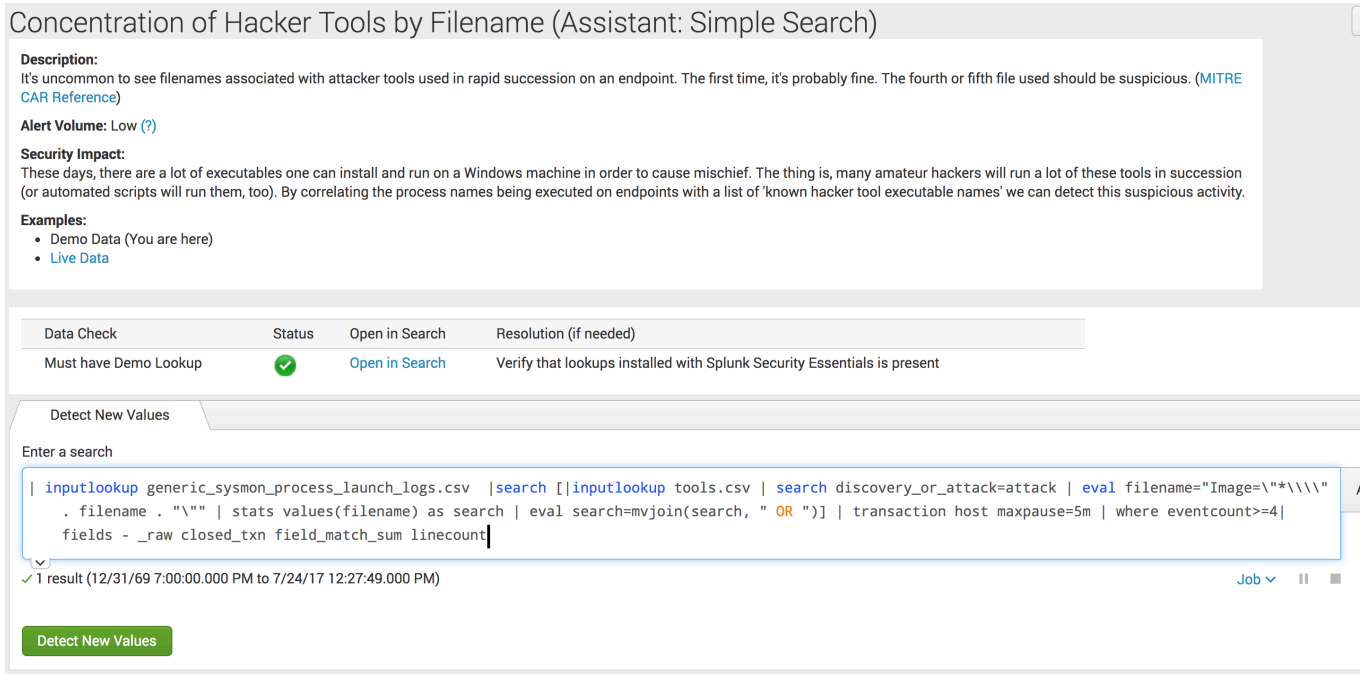


# Search: When Log Sources Go Quiet

## Working Example

- In Splunk Security Essentials we have any example of the earlier query

1. Download the app off Splunkbase
2. Open up Hosts Where Security Sources Go Quiet
3. Click "Show SPL" to see the SPL




splunk> .conf2017

# Key Takeaways

This is where the subtitle goes

1. Watch the earlier Ninjutsus when you get home: dvsplunk.com or conf.splunk.com

2. Grab the PDF Version of this deck and dig in deeper
   Hey, you're on the PDF version. Look at you, ahead of the game! You should go watch the video though - conf.splunk.com 5-6 weeks after conf.

3. Grab the app(s) and explore examples

# Thank You

Don't forget to rate this session in the .conf2017 mobile app

splunk> .conf2017

I get to come back if you give me good ratings. Rate high, early, and often!