Security Ninjutsu Part Five

Our SPL goes to 12. Yeah, 11 isn't enough.

David Veuve | Principal Security Strategist

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> .conf18

NOTES

- Talk about "where people typically stop" versus if they pushed further and got to the brave new future

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!

splunk> .conf18

Agenda

- Introductions – Who Am I, Who Are You?
- Past Ninjutsus
- What Problem Are We Trying to Solve and How?
- Step 1: Build a Dataset
- Step 2: Build your Analytics
- Step 3: Handle Noisy Metrics

Introductions

Personal Introduction

- **David Veuve**
  Principal Security Strategist, Splunk
- dveuve@splunk.com
- Former Splunk Customer
- **Apps:**
  - **Splunk Security Essentials**
  - SplunkJS For All
  - Search Activity
  - Newsletter
  - Three more

- **2018 Talks:**
  - Security Ninjutsu Part Five: The Most Advanced Content Money Can Buy
  - Splunk Security Essentials: What's New and What's Awesome
  - Go From Dashboards to Applications With Ease: SplunkJS for Non-Developers
- **Past Conf Experience**
  - 8 Talks
  - Delivered 11 Times
  - To 2800+ people

splunk> .conf18

Who Are You?

You're New To Splunk (or Security)

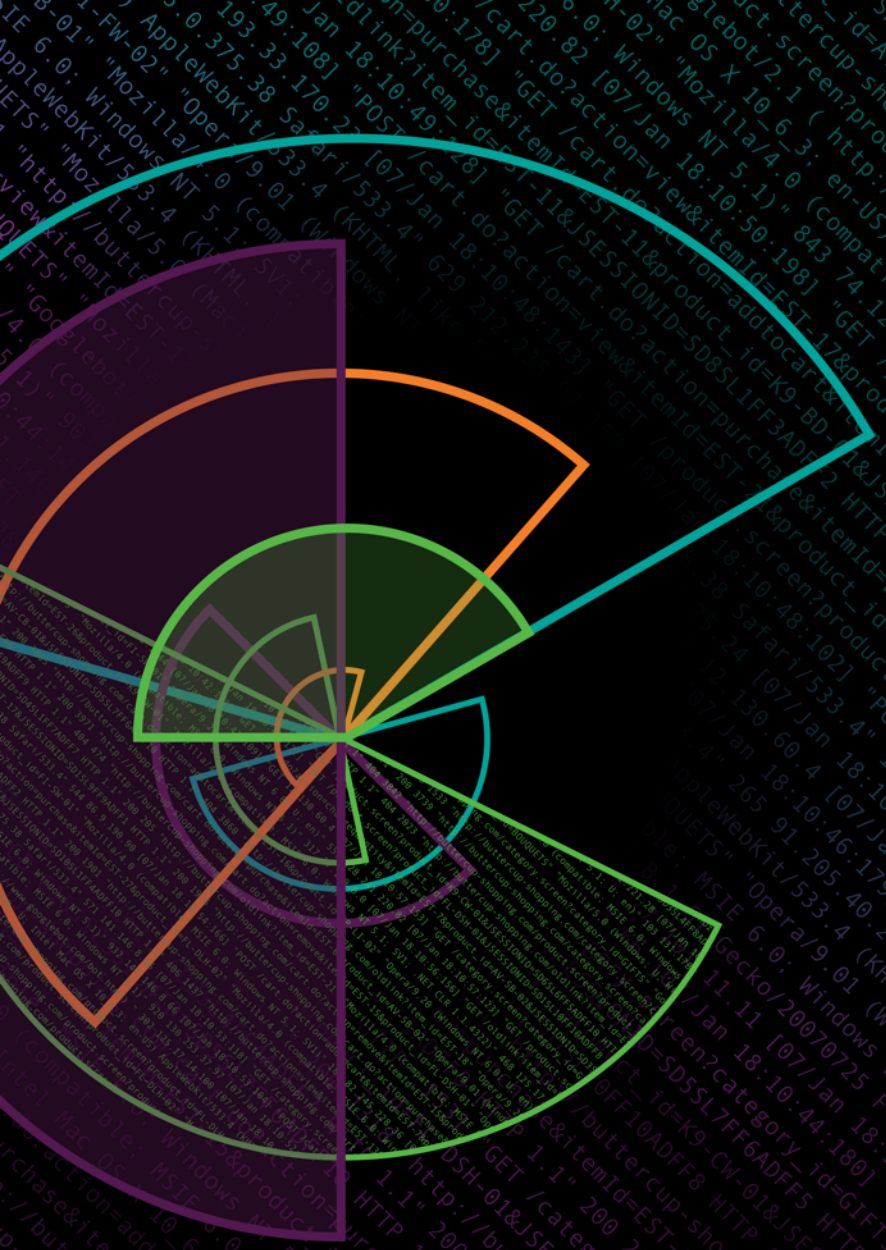
- You like to jump into the deep end
- You want sweet SPL
- You want to get to value quicker

You're Experienced with Splunk

- You want to hear what has made other customers successful
- You want the fanciest of SPL
- You want to get value quicker

Want the Cheat Codes?

- Try out Splunk Security Essentials
- Pre-built content, and usage of the techniques discussed in this session
- 125+ Correlation Searches
- http://apps.splunk.com/app/3435



Past Ninjutsus

Ninjutsus Part One to Three

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 this year.
https://www.davidveuve.com/splunk.html

splunk> .conf18

Ninjutsu Part Four: All the SPL I Know

- Intermediate
  - Common Information Model
  - eval
  - Multi-Value Fields
  - stats
  - stats on stats
  - Formatting a Table
  - Multi-Scenario Alerts
  - Inline Comments
  - Tuning
  - Stats+eval
  - Override Urgency / Severity / Risk
  - Common Apps
  - Risk
  - Subsearches
- Advanced
  - Summary Indexing
  - Lookup Caching
  - Confidence Checking
  - Managing Alert Fatigue
  - Transaction (in a good way)
  - First Time Seen Detection
  - Time Series Detection
  - Time Series + First Time Seen Detection
- Ninja
  - tstats
  - Timestamps & Timestamps
  - Advanced Search Commands
  - Metacharacteristics
  - Machine Learning Toolkit Numeric Clustering
  - Approach to Analytics
- End-to-End
  - When Log Sources Go Quiet

There's even more valuable content in Ninjutsu Part Four. This is the largest volume of content on advanced SPL I've seen or created. https://www.davidveuve.com/splunk.html

splunk> .conf18

What Problem are we Trying to Solve and How?

The Problem

- Many customers with sufficient knowledge of SPL and Splunk never take advantage of the analytics capabilities
- They focus on the analytics themselves and get frustrated by complexity or scale issues
- But their core problem is not the analytics – they don't come to the data with a framework for scaling to build advanced detections
- I routinely deploy many advanced detections – it's not because I'm better at analytics, it's because I build a foundation that makes the detections easy
- I want to teach you that foundation
- I also want to teach you a bunch of tips and tricks about SPL that will make you powerful even if you never do the rest of this

Given a new log, it should not take days to build:

5+ patterns/rules

10+ anomaly detections

1 machine learning detection

…and make it all useful

Three Steps

1. Build a dataset first
   - Scale that dataset to months of data
   - Leverage Summary Indexing / Data Model Acceleration
2. Build your analytics by leveraging patterns as much as possible
   - Additional Reading: Security Ninjutsu Part Four – https://www.davidveuve.com/splunk.html
   - Additional Reading: SEC1583 - Turning Security Use Cases Into SPL
3. Use risk to detect subtle threats
   - Additional Reading: Security Ninjutsu Part Two and Four – https://www.davidveuve.com/splunk.html
   - Additional Reading: SEC1479 - Say Goodbye to Your Big Alert Pipeline, and Say Hello to Your New Risk-Based Approach

Example Scenario

- Suppose that we wanted to detect public AWS buckets, and a get logs in case we needed to investigate an incident
- Management gives us the budget, we get AWS Cloudtrail in, and we're happy
- Then management tells us that we need to do more to justify what we just spent. (Or maybe you just want to do more to secure your environment.)

- Where should you start?
  - Splunk Security Essentials has 10 Cloudtrail-based detections
  - Enterprise Security Content Update has 22 Cloudtrail-based detections

- What else, though?

Step One
Building a Dataset

What is a Dataset

- Any table of data that you can do additional analytics on
  - Could be a data model
  - Could be | table field1 field2 field3
  - Could be a | stats avg(count) min(count) max(count)…

- What is *not* a dataset
  - A bunch of raw data
  - A search that can't scale (well, it's a dataset, it's just a bad one for our purposes)

General Approach to Building a Dataset

- Look at the fields available in some data
- Think of what fields will be interesting
- Think of different ways in which those fields might end up being used
- Scale that dataset up with Data Model Acceleration or Summary Indexing

- Let's look at our CloudTrail data as an example….

An Incomplete List of Useful Fields in CloudTrail Logs

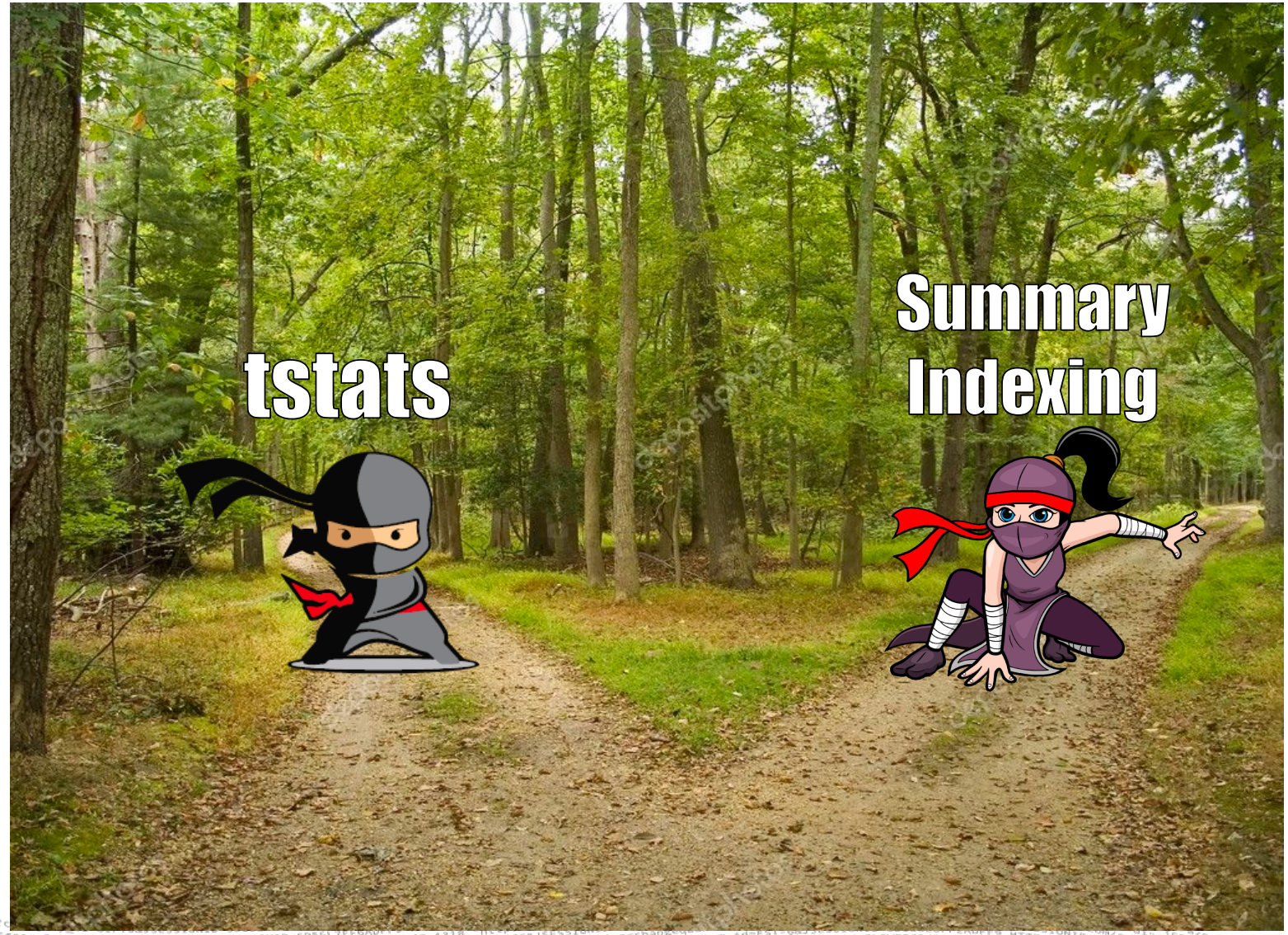
- APIs called – values(), # write, # read, # high risk, values(highrisk()), dc(), count()
  - Examples: PutBucketACL, RunInstances, and DescribeInstances
- Region called – values(), maybe # per standard region, dc()
  - Examples: us-east-1, us-west-2
- src_ip – values(), values(iplocation data), dc(), is_in_aws?
  - Includes the IP address that the API call was made from (warning: could be other AWS instances because life is tricky)
- Instance IDs – dc(), maybe values() if you don't create many or if it's important
  - The unique identifier of the Instances being acted upon
- Bucket Names – dc(), values(), values(permissions)
  - The display name of the buckets being acted upon
- AccessKeyID – dc()
  - All AWS API calls use a key. Sometimes this is a long-lived key, sometimes it's very short lived, but regardless it will be stored here.
- userAgent – values(), dc()
  - The user-agent of the device making the call (frequently aws boto, but can vary)
- mfaAuthenticated – count(true), count(false)
  - Was multi-factor authentication used?
- errorCode – values(), count(), dc()
  - Was there an error code?
- _time - # during working day, # during morning, # overnight
  - When did this occur?
- userIdentity.arn
  - Who took this action?

- Wouldn't it be useful if we could do whatever analysis we wanted on any of these fields, or any of the aggregations of them, e.g.: dc(Bucket Name) per user per day
- Wouldn’t this open all kinds of possibilities?
- It turns out to be not that hard to build this out (it took me about 1 hour.. don't worry, I'll give you the SPL)
- But a dataset is only as good as its scalability, so let's look at how we scale this.

splunk> .conf18

Two Paths to Scale This Dataset

Both Excellent



Path One: tstats

What is tstats?

- The real powerhouse here is Data Model Acceleration (DMA)
- tstats is a search command that allows you to search data that is accelerated in a data model acceleration.
  - Why call this path tstats if data model acceleration is doing the real work? tstats sounds cooler and has fewer syllables.
- DMA will look at the raw data, pull out pre-configured fields, do some streaming evals and lookups if desired, and put them in a tsidx file so that you can search them at extremely high speed with tstats
  - How fast? Fastest I've personally witnessed was 11,000x. Aka 1,100,000% faster. So.. pretty fast.

Will tstats work?

Pros

- Easy
- Can be accelerated almost instantly
- Keeps full data (you only lose milliseconds on the timestamps)
- Allows questions like "show me what operations were taken on instance i-30528350"
- Pivot interface is user friendly
- Recent data always searchable

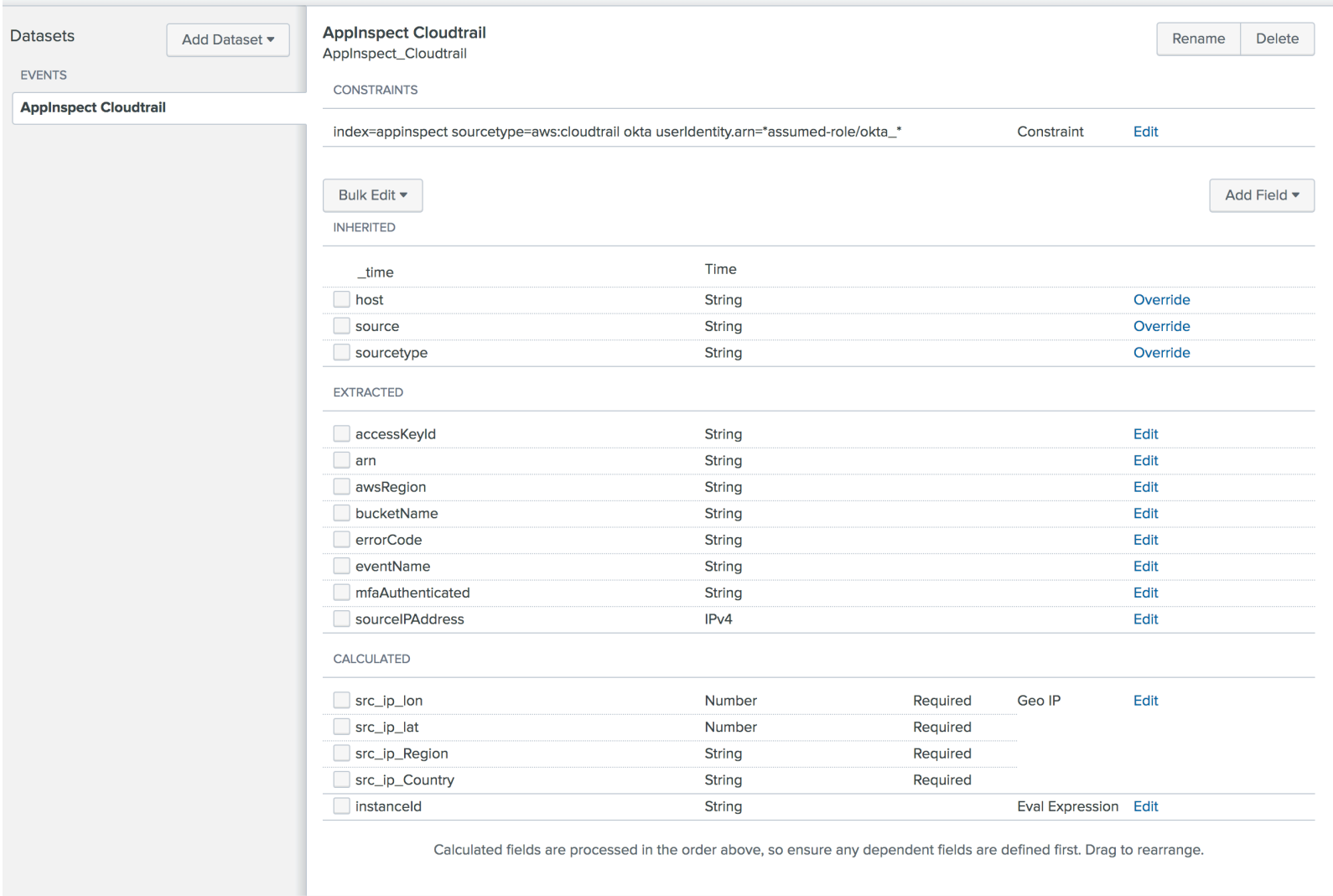
Cons

- Analysts will rarely use it (only available in a dashboard, pivot, or | tstats)
- Storage is higher
- Can't embed eval statements in your tstats, e.g.: | tstats count(eval(awsRegion="us-east-1")) as east ... **does not work**
- Can run into obnoxious limitations in high scale environments because you can't use embedded eval statements
- tstats syntax is weird at first, and has a few "design surprises"

splunk> .conf18

tstats – Best for Unplanned Queries + Dashboards

- It enables you to ask many unexpected queries and drill down into small details
- In Splunk Enterprise Security, most dashboards are powered by tstats
- It does meet *most* detection needs, though there are gaps



splunk> .conf18

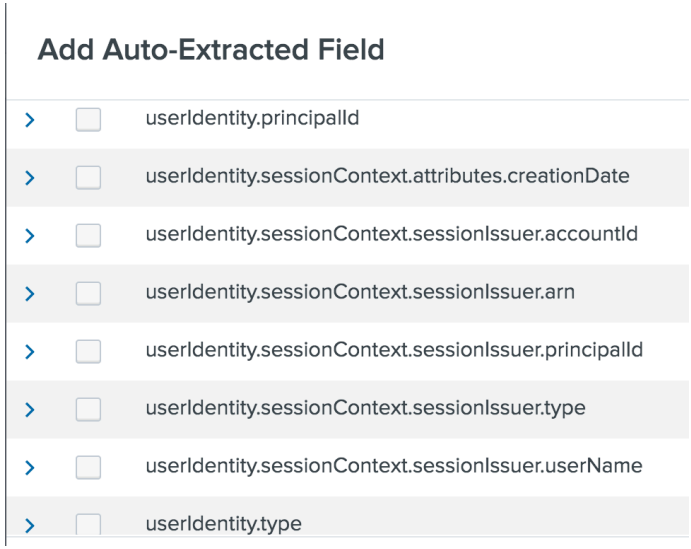
Techniques to be Successful with tstats

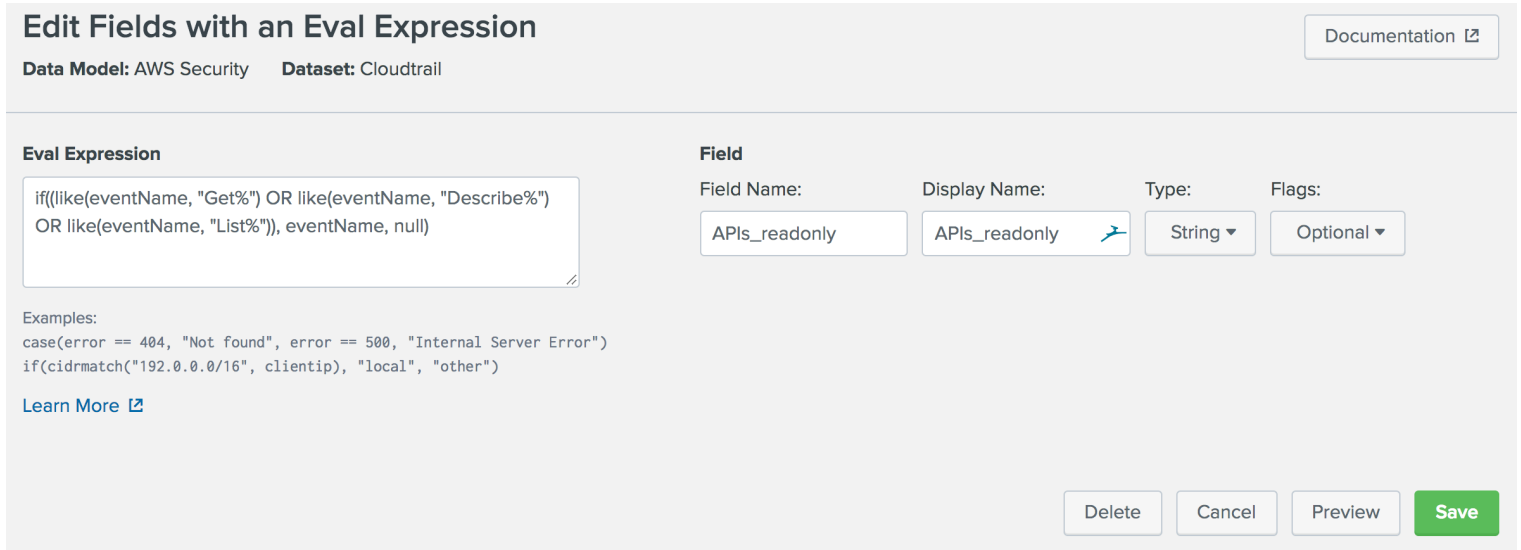
- summariesonly=t allow_old_summaries=t
  - Leverage this to only search accelerated data, making long term searches return very quickly at the expense of potentially missing a few minutes of data.
- datamodelsimple
  - Without datamodelsimple it's horrendously difficult to figure out what fields are actually in a data model, and thus actually use your search. This search command ships with the Common Information Model app
- Add calculated fields for your eval fun
  - Many of the limitations of missing embedded eval statements can be made up for by adding calculated fields (evals) in your data model definition
- Sometimes the fields that show up in the UI aren't actually in the data model. I don't know, but | datamodelsimple solves that

- tstats in general is summarized from slide 136 in last year's Security Ninjusu Part Four, but I also made an entire presentation on this! (Presented at .conf2017 and .conf2016) https://www.davidveuve.com/splunk.html#tstats

Building Your Data Model

- Go into the Data Models config and click new
- You can add all of those fields we discussed before:
- You can also create eval (calculated) fields





Sample Data Model

- If you've got good eyes.. This is the example data model created while building out this talk
- If you don't have good eyes, or want to know what's behind those calculated fields.. Maybe download it instead:
- https://www.davidveuve.com/talks/ninjutsu-part-five/SA_sample_aws_security.spl
- Not intended to be perfect, but maybe a starting point

Datasets | Add Dataset

EVENTS

Cloudtrail

Cloudtrail
Cloudtrail

CONSTRAINTS

index=aws sourcetype=aws:cloudtrail	Constraint

Bulk Edit

INHERITED

_time	Time	
host	String	
source	String	
sourcetype	String	

EXTRACTED

accessKeyId	String	
arn	String	Hidden
awsRegion	String	
bucketName	String	
errorCode	String	
eventName	String	
mfaAuthenticated	String	
sourceIPAddress	IPv4	

CALCULATED

src_ip_lon	Number	Required	Geo IP
src_ip_lat	Number	Required	
src_ip_region	String	Required	
src_ip_country	String	Required	
instanceId	String		Eval Expression
APIs_readonly	String		Eval Expression
APIs_edit	String		Eval Expression
APIs_highrisk	String		Eval Expression
awsRegion_AMER	String		Eval Expression
awsRegion_nonAmer	String		Eval Expression
src_ip_latlon	String		Eval Expression
count_no_mfa	Number		Eval Expression
count_with_mfa	Number		Eval Expression
accessKeyId	String		Eval Expression
bucketName	String		Eval Expression
arn	String		Eval Expression
mfaAuthenticated	String		Eval Expression

Calculated fields are processed in the order above, so ensure any dependent fields are defined first. Dr

Data Model != Complete

- Okay, we have a data model.. but that's not the dataset I promised earlier, with all of the distinct counts and such.
- Fortunately we can do subsequent searches on top of the data model acceleration at high speed (with tstats!)
- While you would probably usually include all of the fields…
  you certainly could…

Build Our Dataset with tstats

| tstats

count as NumEventsOverall

values(Cloudtrail.APIs_edit) as WriteAPINames count(Cloudtrail.APIs_readonly) as ReadAPIs count(Cloudtrail.APIs_edit) as WriteAPIs count(Cloudtrail.APIs_highrisk) as HighRiskAPIs dc(Cloudtrail.eventName) as NumAPIs

values(Cloudtrail.awsRegion) as Regions dc(Cloudtrail.awsRegion_AMER) as AMERRegions dc(Cloudtrail.awsRegion_nonAmer) as NonAMERRegions dc(Cloudtrail.awsRegion) as NumRegions

values(Cloudtrail.sourceIPAddress) as src_ip values(Cloudtrail.sourceIPAddress_Country) as src_ip_country values(Cloudtrail.sourceIPAddress_Region) as src_ip_region values(Cloudtrail.src_ip_latlon) as src_ip_latlon dc(Cloudtrail.sourceIPAddress) as NumSrcIP

values(Cloudtrail.instanceId) as instanceId dc(Cloudtrail.instanceId) as NumInstances

values(Cloudtrail.bucketName) as bucketNames dc(Cloudtrail.bucketName) as numBuckets

dc(Cloudtrail.accessKeyId) as numAccessKeyIds

sum(Cloudtrail.count_no_mfa) as requestsWithoutMFA sum(Cloudtrail.count_with_mfa) as requestsWithMFA

values(Cloudtrail.errorCode) as errorCodes count(Cloudtrail.errorCode) as totalErrorCount dc(Cloudtrail.errorCode) as NumErrorCodes

from datamodel=AWS_Security by Cloudtrail.arn _time span=1d

splunk> .conf18

… but…

- I had to add 8 fields to my data model as I went through to make that query:
  - count_no_mfa
  - count_with_mfa
  - src_ip_latlon
  - awsRegion_AMER
  - awsRegion_nonAmer
  - APIs_readonly
  - APIs_edit
  - APIs_highrisk
- With normal Splunk searching, you can use stats+eval to create these on the fly.
- That's not possible with tstats, which is a big limitation

Path Two: Summary Indexing

splunk> .conf18

Will Summary Indexing Work?

Pros

- Creates Native Events that are easy to search
- Summarizes Data (in one test, 800:1 summarization)
- All the flexibility of stats + eval
- Can retain beyond data retention

Cons

- Lose the ability to run detailed queries
- Not as easy as data model acceleration to manage
- Latency – recent data not searched

Basic Premise of Summary Indexing

- The search head runs some search, say sourcetype=aws:cloudtrail | stats count by user
- This would result in a series of rows on the screen with a count and a username. You're with me so far.
- If we add | collect index=aws_summaries to the end of that: sourcetype=aws:cloudtrail | stats count by user | collect index=aws_summaries
- Then the search head will do the same thing, and write out a special format file to $SPLUNK_HOME/var/spool/splunk with each row on a new line starting with a timestamp and then a string of key-value pairs.
- The indexing process will then read file in, and send it according to your outputs.conf
  - You do have outputs.conf pointed to your indexers, right?
  - If you don't, press the pause button on this PDF and go fix your SH outputs.conf.
- That data is now all indexed and searchable, just like normal data, but at no license cost!

Techniques to be Successful with Summary Indexing

- stats+stats
  - Slide 29 in Ninjutsu Part Four – https://www.davidveuve.com/splunk.html
- stats+eval
  - Slide 48 in Ninjutsu Part Four – https://www.davidveuve.com/splunk.html
- Summary indexing
  - Slide 69 in Ninjutsu Part Four – https://www.davidveuve.com/splunk.html

Prime Directive of Summary Indexing

- Add anything you want before the "by" – it's all pretty cheap on disk space
- Anything after the "by" is *very* expensive
- A low cardinality field after the "by" can be okay.. Adding success/failure *only* doubles your disk space
- A high cardinality field after the "by" will break everything. Adding by eventName will explode event count by 25x in our lab environment, and 9.5x in bytes consumed

KB	# Rows	Search
100	72	sourcetype=aws:cloudtrail | bucket _time span=1d | stats [... Several other fields …] by arn _time
102	72	sourcetype=aws:cloudtrail | bucket _time span=1d | stats **values(eval(if(NOT (like(eventName, "Get%") OR like(eventName, "Describe%") OR like(eventName, "List%")), eventName, null))) as WriteAPINames** [... Several other fields …] by arn _time
136	72	sourcetype=aws:cloudtrail | bucket _time span=1d | stats **values(eventName) as APINames** [... Several other fields …] by arn _time
1282	1836	sourcetype=aws:cloudtrail | bucket _time span=1d | stats [... Several other fields …] by **eventName** arn _time

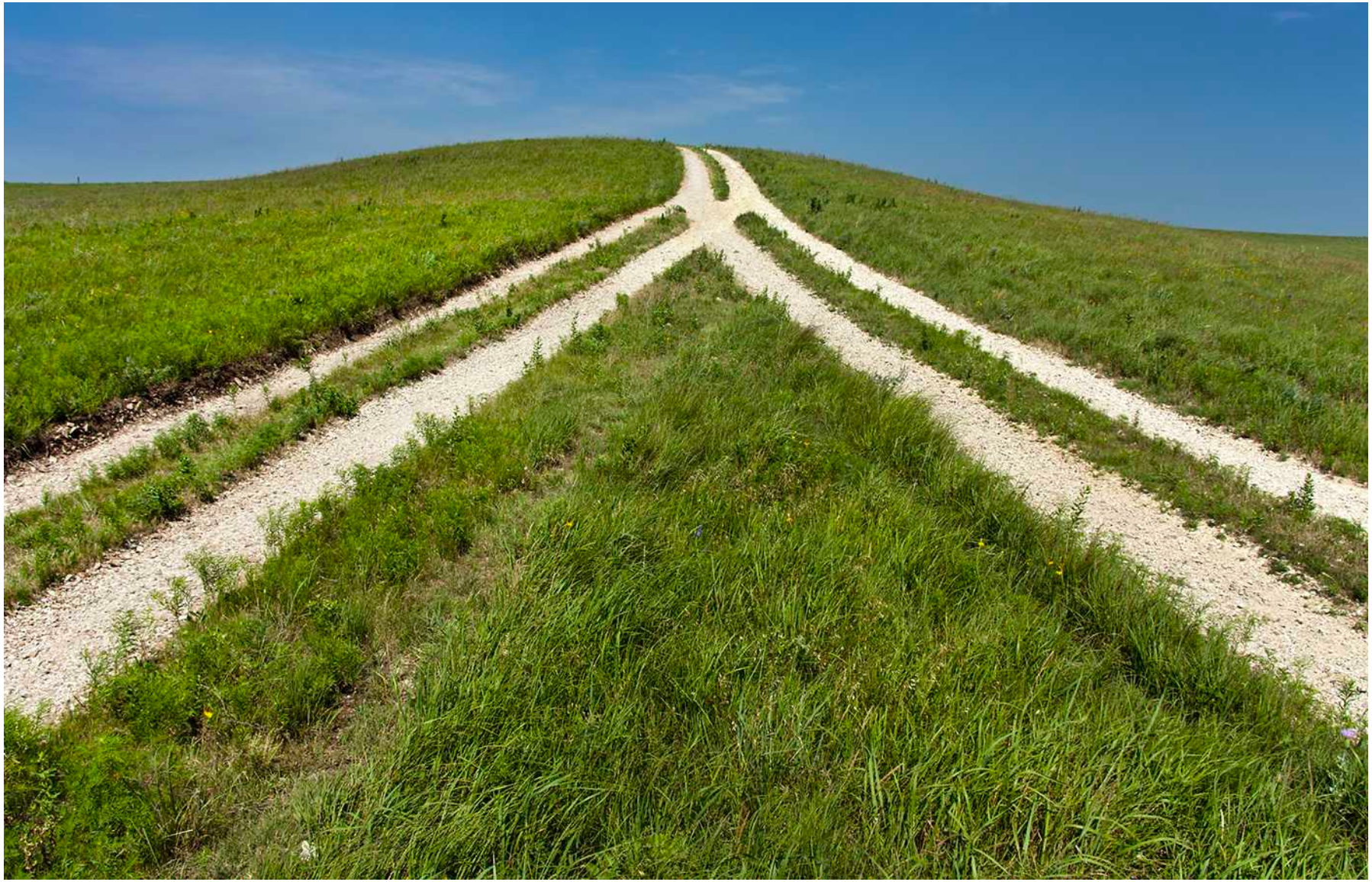
Building Your Summary Index

- index=appinspect sourcetype=aws:cloudtrail okta userIdentity.arn=*assumed-role/okta_*
- | eval sourceIPAddress=mvfilter(sourceIPAddress!="ec2-frontend-api.amazonaws.com")
- | iplocation sourceIPAddress | eval latlon = lat . "," . lon
- | bucket _time span=1d
- | stats
- count as NumEventsOverall
- values(eval(if(NOT (like(eventName, "Get%") OR like(eventName, "Describe%") OR like(eventName, "List%")), eventName, null))) as WriteAPINames count(eval(like(eventName, "Get%") OR like(eventName, "Describe%") OR like(eventName, "List%"))) as ReadAPIs count(eval(NOT (like(eventName, "Get%") OR like(eventName, "Describe%") OR like(eventName, "List%")))) as WriteAPIs count(eval(like(eventName, "Terminate%") OR like(eventName, "RunInstances%") OR like(eventName, "PutBucketPolicy"))) as HighRiskAPIs dc(eventName) as NumAPIs
- values(awsRegion) as Regions dc(eval(like(awsRegion, "us-%") OR like(awsRegion, "ca-%"))) as AMERRegions dc(eval(NOT like(awsRegion, "us-%") OR like(awsRegion, "ca-%"))) as NonAMERRegions dc(awsRegion) as NumRegions
- values(sourceIPAddress) as src_ip values(Country) as src_ip_country values(Region) as src_ip_region values(latlon) as src_ip_latlon dc(sourceIPAddress) as NumSrcIP
- values("responseElements.instancesSet.items{}.instanceId") as instanceNames dc("responseElements.instancesSet.items{}.instanceId") as NumInstances
- values("requestParameters.bucketName") as bucketNames dc("requestParameters.bucketName") as numBuckets
- dc(userIdentity.accessKeyId) as numAccessKeyIds
- count(eval('userIdentity.sessionContext.attributes.mfaAuthenticated' == "false")) as requestsWithoutMFA count(eval('userIdentity.sessionContext.attributes.mfaAuthenticated' != "false")) as requestsWithMFA
- values(errorCode) as errorCodes count(errorCode) as totalErrorCount dc(errorCode) as NumErrorCodes
- by userIdentity.arn, _time

- This will generate our complete rendered dataset
- Summarized per user, per day
- In our test dataset, it summarizes 56k records (20 MB on disk) into 72 records.
- More than a 750x speed improvement! Now that scales!

splunk> .conf18

Two Paths Converge



The Summary Index is *in* the Data Model

- First build a summary index
- Then build a data model on top of that summary index
- Then accelerate that data model

- Query BILLIONS of records in *seconds*

splunk> .conf18

Size on Disk vs Performance

5 users, 45 days of AWS data, 56k requests

Storage	Disk Space	Speed for Query
Raw Data Estimate (w/o Indexed Extractions)	20 MB	7 s
Summary Index	172 kb	0.329 s
Data Model	1402 kb (requires 20 MB raw)	0.317 s
Summary Index + Data Model	207 kb	0.056 s

- Test Query: tracked users who made more high risk API calls than their baseline

"Just one more thing"

JSON Fields with Indexed Extractions

- Technically, I would have been able to do a *lot* of that without the data model acceleration. (Shock!)
- Why? Because on my test system I ingested a test set of logs with sourcetype=json
- sourcetype=json defaults to turning on indexed extractions, which is a feature added in 6.0. With Indexed Extractions, we automatically index every field in JSON, CSV, etc data.
- Indexed Extractions are pretty cool because they searches superfast, and you can actually use tstats directly!
  | tstats count where index=aws sourcetype=aws:cloudtrail by userIdentity.arn
- Fortunately for the integrity of this talk, Indexed Extractions are turned off by default in the AWS app
- Why? Because the explode disk space used on disk
- Not that relevant for AWS CloudTrail, but maybe good to know for your log types.

Step Two
Build Your Analytics

And Leverage Patterns as much as Possible

We've Got Our Dataset – Let's Detect

- Now that we have a dataset, we can easily run a variety of detections
- While many folks can take this dataset and run with it.. It's best to not rebuild the wheel
- There are generally a few different SPL patterns that people will use and repeat many times
- We will walk through four in this section, but there are more
  - Highly recommend attendance: SEC1583 - Turning Security Use Cases Into SPL @ .conf18

- Enough with the preamble and context – let's get to the detections!

Four Types of Patterns Discussed Here

- Basic Patterns / Signatures (You're doing this today)
- Rarity Detections
- Time Series Spikes
- Multi-variate Time Series Clustering w/ MLTK (oh my!)

Examples

- For each of these patterns, for each search style (raw logs, tstats, and summary indexes), we'll provide a couple of examples of actually applying the search
- We promise that these examples aren't exclusive to that search style (e.g., you can solve the same problem we're showing with a tstats search using a summary index)..
- We're only not repeating examples so that we can give you more examples of analytics. (Where we repeat examples is either accidental, or because deadlines are hard sometimes and we ran out of time!)
- There are so many analytics that you could easily deploy!

Basic Patterns and Signatures

- You already know how to do this
- This is the majority of correlation rules in existence

Basic Patterns and Signatures

- We only do have instances in the US and Canada, so we shouldn't have anything beyond those
  - sourcetype=aws:cloudtrail awsRegion!=us-* AND awsRegion!=ca-*
- All Instance Creation in Prod should be done by service accounts
  - sourcetype=aws:cloudtrail userIdentity.accountId=987654321234 userIdentity.arn!=*/automation
- Alert on all public AWS buckets
  - Copy-paste from Splunk Security Essentials: Public S3 Bucket in AWS
  - sourcetype=aws:cloudtrail AllUsers eventName=PutBucketAcl
    | spath output=userIdentityArn path=userIdentity.arn
    | spath output=bucketName path="requestParameters.bucketName"
    | spath output=aclControlList path="requestParameters.AccessControlPolicy.AccessControlList"
    | spath input=aclControlList output=grantee path=Grant{} | mvexpand grantee | spath input=grantee
    | search "Grantee.URI"=*AllUsers
    | table _time, Permission, Grantee.URI, bucketName, userIdentityArn | sort - _time

splunk> .conf18

We Can Also Run Those on our Summary Index

It will just be much much faster

- We only do have instances in the US and Canada, so we shouldn't have anything beyond those
  - index=aws_summary awsRegion!=us-* AND awsRegion!=ca-*
- All Instance Creation in Prod should be done by service accounts
  - index=aws_summary userIdentity.arn!=*/automation
- Alert on all public AWS buckets
  - Copy-paste from Splunk Security Essentials: Public S3 Bucket in AWS
  - Okay, this one we can't actually do with our summary index because it requires fields not in the summary index. Specifically, we didn't extract the Bucket ACL because all that spath'ing is a hassle. It's absolutely possible, just rough in a .conf talk

Rarity Detections

- I want to detect the first time "this" occurs
- I want to detect if "this" occurs less than 1 time out of 20,000 and the last time was at least a day ago
- I want to detect if a user did "this" for the first time

First Time Seen Detections

Simple and Effective

- <datasource> | stats min(_time) as earliest max(_time) as latest by <monitored> | where earliest > relative_time(now(), "-1d@d")
- I want to detect the first instance creation per region
  - **sourcetype=aws:cloudtrail** eventName=runInstances| stats min(_time) as earliest max(_time) as latest by **awsRegion** | where earliest > relative_time(now(), "-1d@d")
- I want to detect activity for the first time per user per region
  - sourcetype=aws:cloudtrail | stats min(_time) as earliest max(_time) as latest by awsRegion, **userIdentity.arn** | where earliest > relative_time(now(), "-1d@d")

First Time Seen Detections – Backed by tstats

- | tstats summariesonly=t allow_old_summaries=t
  min(_time) as earliest max(_time) as latest from datamodel=<..> by <..>
  | where earliest > relative_time(now(), "-1d@d")
- I want to detect when a new Country (based on src_ip) becomes active
  - | tstats summariesonly=t allow_old_summaries=t min(_time) as earliest max(_time) as latest from datamodel=**Example_AWS_Security** by **cloudtrail.sourceIPAddress_Contry** | where earliest > relative_time(now(), "-1d@d")
- I want to detect when users take high risk actions from a new Country (based on src_ip)
  - | tstats summariesonly=t allow_old_summaries=t min(_time) as earliest max(_time) as latest from datamodel=Example_AWS_Security **where cloudtrail.HighRiskAPICalls>0** by cloudtrail.sourceIPAddress_Contry | where earliest > relative_time(now(), "-1d@d")

First Time Seen Detection – Backed by Summary Index

This will have the same format as the raw logs, just faster. (Because Summary Indexes are easy)

- index=<index> | stats min(_time) as earliest max(_time) as latest by <monitored> | where earliest > relative_time(now(), "-1d@d")
- I want to detect the first instance creation per region
  - **index=aws_summary eventName=runInstances**| stats min(_time) as earliest max(_time) as latest by **awsRegion** | where earliest > relative_time(now(), "-1d@d")
- I want to detect activity for the first time per user per region
  - **index=aws_summary** | stats min(_time) as earliest max(_time) as latest by awsRegion, **userIdentity.arn** | where earliest > relative_time(now(), "-1d@d")

splunk> .conf18

Unusual Detection

- <datasource> earliest=-30d@d
  | stats count latest(_time) as latest by <monitored> [optionally: <entity>]
  | eventstats sum(count) as total [optionally: by <entity>]
  | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")
- I want to detect rare API Calls (eventName)
  - sourcetype=aws:cloudtrail earliest=-30d@d | stats count by **eventName** | eventstats sum(count) as total | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")
- I want to detect users making rare API Calls
  - sourcetype=aws:cloudtrail earliest=-30d@d | stats count by eventName, **userIdentity.arn** | eventstats sum(count) as total by **userIdentity.arn** | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")

Unusual Detection – Backed by tstats

- | tstats count latest(_time) as latest from datamodel=<…> where earliest=-30d@d by <monitored>
  | eventstats sum(count) as total
  | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")
- I want to detect unusual errors
  - | tstats count latest(_time) as latest from datamodel=**Example_AWS_Security** where earliest=-30d@d by **cloudtrail.errorCode** | eventstats sum(count) as total | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")
- I want to detect users with an unusual MFA Status
  - | tstats count latest(_time) as latest from datamodel=**Example_AWS_Security** where earliest=-30d@d by **cloudtrail.mfaAuthenticated cloudtrail.userIdentity.arn** | eventstats sum(count) as total | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")

Unusual Detection – Backed by Summary Index

This will have the same format as the raw logs, just faster. (Because Summary Indexes are easy)

- index=<index> earliest=-30d@d
  | stats count latest(_time) as latest by <monitored> [optionally: <entity>]
  | eventstats sum(count) as total [optionally: by <entity>]
  | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")
- I want to detect rare API Calls (eventName)
  - index=aws_summary earliest=-30d@d | stats count by **eventName** | eventstats sum(count) as total | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")
- I want to detect users making rare API Calls
  - index=aws_summary earliest=-30d@d | stats count by eventName, **userIdentity.arn** | eventstats sum(count) as total by **userIdentity.arn** | where count / total < 1/20000 AND latest > relative_time(now(), "-1d@d")

How Long is Your Baseline?

- How long you want to maintain this baseline (how far back to search) can vary from organization to organization
- Frequent time ranges are between 30 days and 100 days
- Beyond 100 days, you over-value the past (whatever my job was 5 months ago matters little in 2018)
- Less than 30 days, you don't have enough confidence
- This will apply to the next section (time series) as well

Time Series Spikes

- Detect if "this" occurs more than it usually does (StDev)
- Has "this" ever occurred more than it typically does (IQR)

splunk> .conf18

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

Time Series Spikes – Correlation Search Style

This will compare the last day to the baseline

- <datasource> | bucket _time span=1d | stats count by <monitored> _time
  | stats latest(count) as latest avg(count) as avg stdev(count) as stdev by <monitored>
  | where latest > avg + 6*stdev
- Detect users launching more instances than usual
  - sourcetype=aws:cloudtrail eventName=runInstances | bucket _time span=1d | stats count by userIdentity.arn _time | stats latest(count) as latest avg(count) as avg stdev(count) as stdev by userIdentity.arn | where latest > avg + 6*stdev
- Detect users modifying S3 bucket ACLs more than usual
  - sourcetype=aws:cloudtrail eventName=PutBucketACL | bucket _time span=1d | stats count by userIdentity.arn _time | stats latest(count) as latest avg(count) as avg stdev(count) as stdev by userIdentity.arn | where latest > avg + 6*stdev

Ok, It's Actually a Little More Complicated

```
… | 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

Time Series Spikes – Correlation Search Style

Doing the Maths Correctly

- <datasource> | bucket _time span=1d | stats count by <monitored> _time | stats 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 <monitored> | where latest > avg + 6*stdev
- Detect users launching more instances than usual
  - sourcetype=aws:cloudtrail eventName=runInstances | bucket _time span=1d | stats count by userIdentity.arn _time | stats 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 userIdentity.arn | where latest > avg + 6*stdev
- Detect users modifying S3 bucket ACLs more than usual
  - sourcetype=aws:cloudtrail eventName=PutBucketACL | bucket _time span=1d | stats count by userIdentity.arn _time | stats 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 userIdentity.arn | where latest > avg + 6*stdev

splunk> .conf18

Time Series Spikes – Backed by tstats

- | tstats count from datamodel=<datamodel> where earliest=-30d@d by <monitored> _time span=1d
  | stats 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 <monitored>
  | where latest > avg + 6*stdev
- Detect an API being hit more than usual
  - | tstats count from datamodel=**AWS_Security** where earliest=-30d@d by **Cloudtrail.eventName** _time span=1d
    | stats 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 **Cloudtrail.eventName**
    | where latest > avg + 6*stdev
- Detect users touching S3 buckets more than usual
  - | tstats dc(Cloudtrail.bucketName) as count from datamodel=AWS_Security where earliest=-30d@d by **Cloudtrail.userIdentity.arn** _time span=1d
    | stats 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 **Cloudtrail.userIdentity.arn**
    | where latest > avg + 6*stdev

splunk> .conf18

Time Series Spikes – Backed by Summary Indexing

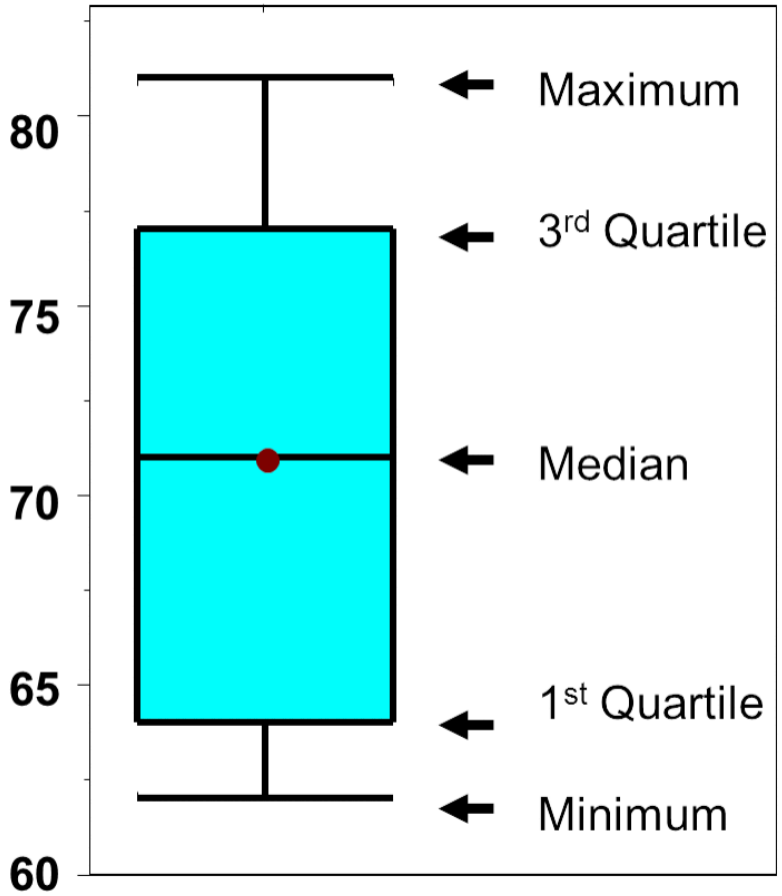
- index=<index>
  | stats max(eval(if(_time >= relative_time(now(), "-1d@d"),<field>, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d@d"), <field>, null))) as avg stdev(eval(if(_time < relative_time(now(), "-1d@d"), <field>, null))) as stdev by <monitored>
  | where latest > avg + 6*stdev
- Detect users running more high risk API operations than usual
  - index=**aws_cloudtrail**
    | stats max(eval(if(_time >= relative_time(now(), "-1d@d"),**HighRiskAPIs**, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d@d"), **HighRiskAPIs**, null))) as avg stdev(eval(if(_time < relative_time(now(), "-1d@d"), **HighRiskAPIs**, null))) as stdev by **userIdentity.arn**
    | where latest > avg + 6*stdev
- Detect more operations in a region than normal
  - index=**aws_cloudtrail | makemv Regions**
    | stats max(eval(if(_time >= relative_time(now(), "-1d@d"),**NumAPIs**, null))) as latest avg(eval(if(_time < relative_time(now(), "-1d@d"), **NumAPIs**, null))) as avg stdev(eval(if(_time < relative_time(now(), "-1d@d"), **NumAPIs**, null))) as stdev by **Regions**
    | where latest > avg + 6*stdev

splunk> .conf18

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 percentile.
- 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.



Historical Spikes

- <datasource> | bucket _time span=1d | stats count by <monitored> | eventstats perc25(count) as perc25 perc75(count) as perc75 by <monitored> | where count > perc75 + (perc75 - perc25) * 1.5
- Something weird happened. Over the last weeks, has anyone run more high risk APIs than usual?
  - sourcetype=aws:cloudtrail | bucket _time span=1d | stats count by <monitored> | eventstats perc25(count) as perc25 perc75(count) as perc75 by <monitored> | where count > perc75 + (perc75 - perc25) * 1.5

Historical Spikes – Backed by tstats

- | tstats count from datamodel=<datamodel> where earliest=-30d@d by <monitored> _time span=1d | eventstats perc25(count) as perc25 perc75(count) as perc75 by <monitored> | where count > perc75 + (perc75 - perc25) * 1.5
- Something weird happened. Over the last weeks, has anyone run more high risk APIs than usual?
  - | tstats **dc(Cloudtrail.HighRiskAPIs) as count** from datamodel=**AWS_Security** where earliest=-30d@d by **Cloudtrail.arn** | eventstats perc25(count) as perc25 perc75(count) as perc75 by **Cloudtrail.arn** | where count > perc75 + (perc75 - perc25) * 1.5
- Something weird happened. Over the last weeks, has anyone touched more buckets than usual?
  - | tstats **dc(Cloudtrail.BucketNames) as count** from datamodel=AWS_Security where earliest=-30d@d by Cloudtrail.arn | eventstats perc25(count) as perc25 perc75(count) as perc75 by Cloudtrail.arn | where count > perc75 + (perc75 - perc25) * 1.5

splunk> .conf18

Historical Spikes – Backed by Summary Indexing

- index=<index> | eventstats perc25(<field>) as perc25 perc75(<field>) as perc75 by <monitored> | where count > perc75 + (perc75 - perc25) * 1.5
- Something weird happened. Over the last weeks, has anyone run more high risk APIs than usual?
  - index=aws_summary | eventstats perc25(**HighRiskAPIs**) as perc25 perc75(**HighRiskAPIs**) as perc75 by userIdentity.arn | where count > perc75 + (perc75 - perc25) * 1.5
- Something weird happened. Over the last weeks, has anyone touched more buckets than usual?
  - index=aws_summary | eventstats perc25(**numBuckets**) as perc25 perc75(**numBuckets**) as perc75 by userIdentity.arn | where count > perc75 + (perc75 - perc25) * 1.5

splunk> .conf18

Multi-variate Time Series Analysis with Clustering

- We discussed this last year at .conf. The basic gist is:

  We're looking at a table full of time series data points per user. I can build out rules for all of these different specific scenarios… how do I detect just general "weird" behavior? Some amount of variation across all of these metrics without stating explicitly what I want?

- Isn't this "Deep Learning"? No, but if you want to invest in the ICO of "DavidCoins" then let me know.

Basic Gist

1. First we normalize everyone to how many standard deviations they are away from their own baseline
   - That way we're not comparing a sysadmin to a casual user, with a very different behavior pattern
2. Next we use PCA to reduce the number of fields
   - PCA stands for Principal Component Analysis and "compresses" data points. Think of it like converting that FLAC audio file to a 384 kbps MP3. You can't really tell which is which!
3. Then we use k-means clustering to group fields
4. Then we use IQR to look for nodes that are far from their cluster, or in a very small cluster by themselves

MLTK Backed by tstats

You'll probably want to tune that to useful fields

- | tstats dc("Cloudtrail.mfaAuthenticated") as "num_Cloudtrail.mfaAuthenticated" dc("Cloudtrail.sourceIPAddress") as "num_Cloudtrail.sourceIPAddress" dc("Cloudtrail.sourceIPAddress_Region") as "num_Cloudtrail.sourceIPAddress_Region" dc("Cloudtrail.sourceIPAddress_Country") as "num_Cloudtrail.sourceIPAddress_Country" dc("Cloudtrail.awsRegion_AMER") as "num_Cloudtrail.awsRegion_AMER" dc("Cloudtrail.userIdentity.accessKeyId") as "num_Cloudtrail.userIdentity.accessKeyId" dc("Cloudtrail.instanceId") as "num_Cloudtrail.instanceId" values("Cloudtrail.count_with_mfa") as "Cloudtrail.count_with_mfa" dc("Cloudtrail.sourceIPAddress_City") as "num_Cloudtrail.sourceIPAddress_City" dc("Cloudtrail.src_ip_latlon") as "num_Cloudtrail.src_ip_latlon" dc("Cloudtrail.errorCode") as "num_Cloudtrail.errorCode" dc("Cloudtrail.requestParameters.bucketName") as "num_Cloudtrail.requestParameters.bucketName" dc("Cloudtrail.APIs_edit") as "num_Cloudtrail.APIs_edit" dc("Cloudtrail.awsRegion_nonAmer") as "num_Cloudtrail.awsRegion_nonAmer" dc("Cloudtrail.APIs_readonly") as "num_Cloudtrail.APIs_readonly" dc("Cloudtrail.awsRegion") as "num_Cloudtrail.awsRegion" from datamodel=AWS_Security by "Cloudtrail.arn" _time span=1d

  | eventstats avg(*) as AVG_* stdev(*) as STDEV_* by "Cloudtrail.arn"
  | 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

splunk> .conf18

You'll probably want to tune that to useful fields

MLTK Backed by Summary Indexing

- index=summary | makemv Regions | makemv src_ip | makemv src_ip_Country | makemv bucketNames | stats values(AMERRegions) as AMERRegions values(HighRiskAPIs) as HighRiskAPIs values(NonAMERRegions) as NonAMERRegions values(Num*) as Num* values(ReadAPIs) as ReadAPIs dc(Regions) as Regions dc(WriteAPINames) as WriteAPIDC values(WriteAPIs) as WriteAPIs dc(bucketNames) as UniqueBucketNames dc(errorCodes) as errorCodes dc(instanceId) as InstanceIDs values(requests*) as requests* dc(src_*) as src_* values(totalErrorCount) as totalErrorCount by Cloudtrail.arn _time

  | eventstats avg(*) as AVG_* stdev(*) as STDEV_* by Cloudtrail.arn
  | 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

Step Three
Use Risk to Detect Subtle Threats

Basic Gist

- A lot of these detections are very noisy.
- You can probably eyeball the use case to determine whether you'd actually want to get notified about it every time it happened.
- What we really need here is some way to track events that are relevant, but that you don't necessarily want to put in front of analysts
- We'll talk through three approaches here:
  - Basic Risk Framework (like in ES)
  - The Jack Crook approach to alerts
  - The O365 team approach to alerts
  - The AFI approach to alerts

The Splunk ES Risk Framework

- For any correlation search in ES, you can run an adaptive response action to create a risk indicator
- Define:
  - Risk Score (1-100)
  - Risk Object (the field that the risk applies to)
  - Risk Object Type (user, system, other)
- That will deposit risky events in index=risk

splunk> .conf18

The O365 Team Approach

Matt Swann is a Principal Engineering Manager at Microsoft, and a great Twitter follow

- Alerts
  - These activities have a significant service impact and are rarely due to benign activity. For example, a new account being granted Domain Administrator privileges would be classified as an alert.
  - An alert immediately generates an escalation / page to be reviewed.
- Atomics
  - These are activities that are significant, and unlikely to be benign, but don't risk the enterprise if they're not responded to in short order. For example, a new local account being created on an important system.
  - All atomics should be reviewed, but doing so can happen in their own time.
- Behavioral
  - These activities may occur due to benign service operations but may also indicate unauthorized activity. An example of a Behavioral indicator is a new process executing that has never been observed across the service.
  - These won't be reviewed on their own, but will show up if grouped with other behaviorals or atomics through the clustering described in the post below.
- Contextual
  - These activities occur very frequently due to benign activity but have forensic value during an investigation. A net.exe process start is one type of Contextual indicator.
  - These would never be reviewed directly on their own, but are available to analysts to provide starting points and situational awareness during an investigations.

This is cribbed from @MSwannMSFT, heavily from this blog post. You follow him on Twitter, right?
https://blogs.technet.microsoft.com/office365security/defending-office-365-with-graph-analytics/

The Jack Crook Approach

Jack Crook is a Principal Incident Investigator @ GE and a great Twitter follow

- There are three types of alerts
- Detections that are fed directly to an analyst as an alert
  - High confidence, traditional SIEM alerts
- Detections that are used for correlation
  - Low fidelity, relatively common
  - Threat Intel can often sit in this bucket
  - When you see enough of these, or enough patterns, you can send an alert (generally to a more senior analyst)
- Detections written to increase visibility
  - Can be built out to provide something to correlate against
  - Most likely coming from some other detection tool (IDS, HIPS, HIDS, Proxy, Sysmon, etc.)

This is cribbed directly from @jackcr's blog post. You follow him on Twitter, right?
http://findingbad.blogspot.com/2018/06/methods-of-detection.html

The AFI Approach

- Why would I ruin the surprise? Go see their talk!

Intermediate | Security, Compliance and Fraud

SEC1479 - Say Goodbye to Your Big Alert Pipeline, and Say Hello to Your New Risk-Based Approach

SCHEDULE Tuesday, Oct 02, 3:30 p.m. - 4:15 p.m.

SPEAKERS

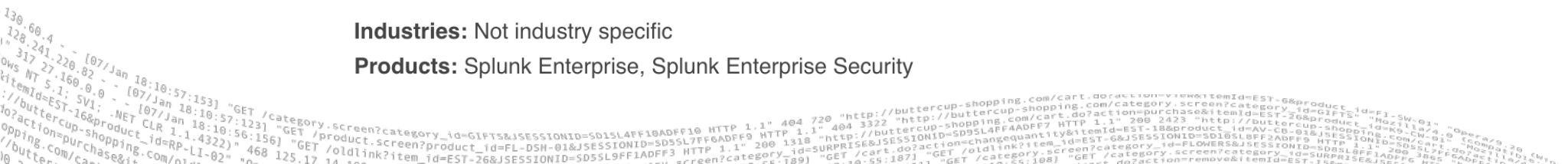
Jim Apger, Staff Security Architect, Splunk

Stuart McIntosh, Cyber Security Threat Specialist, American Family Insurance

A different approach has been evolving within the SOC, which is generally based on risky behavior. The use of the term “generally” is very important and this approach has been quickly spreading from the early adopting mature SOCs to the masses over the past several years. Too many incidents and whitelists? No situational awareness at the start of an investigation? Interested in ATT&CK? The risk-based (or more specifically "risky behavior"-based) approach allows you to deploy more general detection mechanisms without directly creating incidents. These more general risk attributions, by design, are the foundation upon which a new breed of high-confidence incidents is delivered. American Family Insurance will be joined by Splunk to detail its evolution into a risk-based approach within its SOC.

Industries: Not industry specific

Products: Splunk Enterprise, Splunk Enterprise Security



splunk> .conf18

Enough Philosophy – Bring the SPL

splunk> .conf18

Three Initial Steps

1. Collect Risky Events
2. Count users who have a high level of risk
3. Build risk visibility into your playbooks

splunk> .conf18

Collecting Risk Events

- Two approaches.
- If you have ES:
  - Use the Risk adaptive response action. It takes care of everything for you, and it looks great while doing it.
- If you don't have ES:
  - Create an index for storing these risky events
  - Normalize the fields. Something like:
    | eval risk_object=src_ip, risk_object_type="system", risk_score=60
  - Collect the events into your index by adding | collect index=myriskindex
  - E.g.,:
    sourcetype=myIDS | eval risk_object=if(cidrmatch("10.0.0.0/8", src_ip), src_ip, dest_ip) risk_object_type="system", risk_score=case(severity="critical", 30, severity="high", 10, 1=1, 1) | collect index=myriskindex

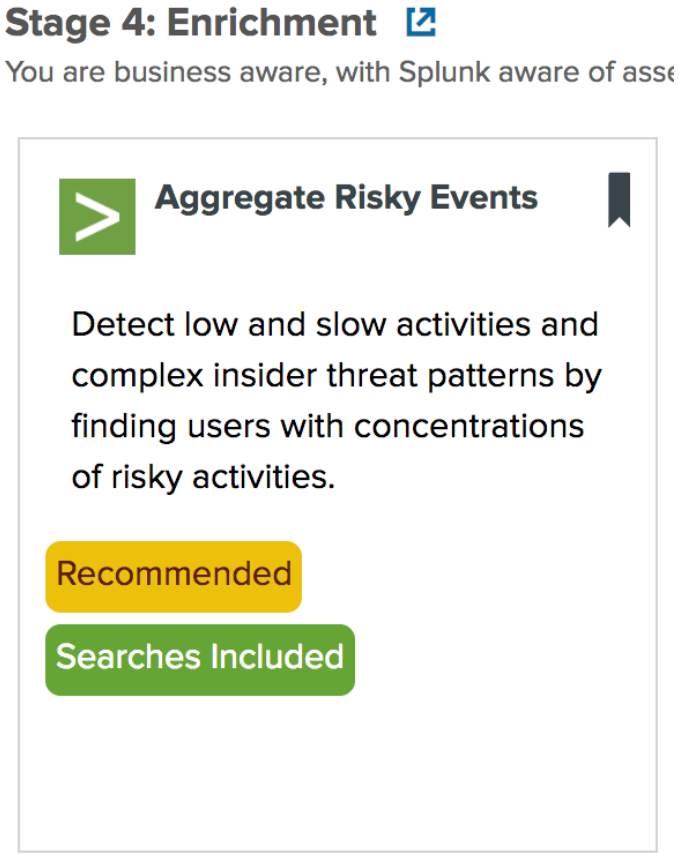
Detecting Risky Users

- Easy mode: go to Splunk Security Essentials 2.2 or above.
- Variant #1: By the number of domains, and number overall

```
index=risk earliest=-7d
| rex field=search_name "^(?<security_domain>[\w ]*) -"
| stats values(security_domain) as security_domain dc(security_domain) as num_security_domain values(search_name) as search_name dc(search_name) as num_search_name by risk_object _time
| where num_security_domain >= 3 OR num_search_name >= 5
```

- Variant #2: By the risk score

```
index=risk earliest=-30d
| stats values(search_name) as search_names sum(risk_score) as thirty_day_risk sum(eval(if(_time > relative_time(now(), "-1d@d"),risk_score,0))) as one_day_risk by risk_object
| eval threshold_1day = 500, threshold_30day = 1200
| where one_day_risk>threshold_1day OR thirty_day_risk>threshold_30day
| 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)
```



splunk> .conf18

On Tuning

- Anytime you're looking at building out threshold detections, expect a lot of tuning and tweaking
- Make sure you have a good relationship with the folks who will be receiving the alerts
  (including yourself – the amount of grief the choices of "last week David" has given me is immense)
- Test the alerts before you feed them into the analyst queue to make sure they're suitably actionable

Build Visibility into Your Playbooks

- There are many approaches here that can apply to your organization

1. Leverage the Risk swimlane in the ES Asset or Identity Investigator
2. Use phantom (or, gasp, other orchestration) to query your risk index
3. Use the Automatic Search Add-on For Splunk https://splunkbase.splunk.com/app/3837/
4. Add a pane to the investigator workbench http://docs.splunk.com/Documentation/ES/5.1.0/Admin/Customizeinvestigations#Create_a_workbench_profile
5. Just add a normal step into the playbook

Also… Seriously…

- Go check this out

Intermediate | Security, Compliance and Fraud

SEC1479 - Say Goodbye to Your Big Alert Pipeline, and Say Hello to Your New Risk-Based Approach

SCHEDULE Tuesday, Oct 02, 3:30 p.m. - 4:15 p.m.

SPEAKERS

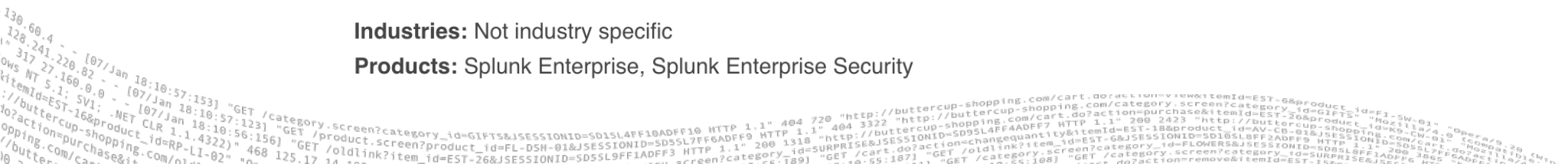
Jim Apger, Staff Security Architect, Splunk

Stuart McIntosh, Cyber Security Threat Specialist, American Family Insurance

A different approach has been evolving within the SOC, which is generally based on risky behavior. The use of the term “generally” is very important and this approach has been quickly spreading from the early adopting mature SOCs to the masses over the past several years. Too many incidents and whitelists? No situational awareness at the start of an investigation? Interested in ATT&CK? The risk-based (or more specifically "risky behavior"-based) approach allows you to deploy more general detection mechanisms without directly creating incidents. These more general risk attributions, by design, are the foundation upon which a new breed of high-confidence incidents is delivered. American Family Insurance will be joined by Splunk to detail its evolution into a risk-based approach within its SOC.

Industries: Not industry specific

Products: Splunk Enterprise, Splunk Enterprise Security



splunk> .conf18

Summary

Limitations

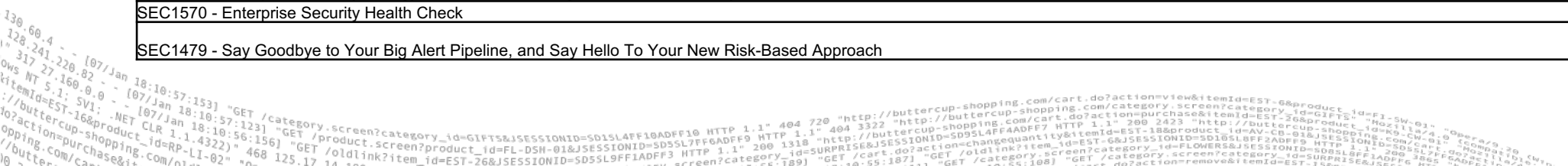
- You should probably not take on super advanced analytics until your cyber house is in order
- If you haven't tried uncomfortably hard to build out an asset list, then go back to that
- If you haven't built out standard patterns, then go back to that
- If you don't have a good concept of what you should be protecting, then go back to that
- But if you've got a reasonable grasp on and are ready to take a step up, welcome to the party.

Steps

- Build a dataset, and scale it out with Summary Indexing or Data Model Accel
- Leverage any of the patterns in this presentation or in others
- Sum up the riskiest entities

Other Recommended Talks

SEC1583 - Turning Security Use Cases into SPL	**Hunting / IR**
SEC1039 - Detection Technique Deep Dive	**Hunting / IR**
SEC1297 - Down in the Weeds, Up in the Cloud: Splunking your Azure and Office 365	Hunting / IR
SEC1355 - Hunting the Known Unknown: Microsoft Cloud	**Hunting / IR**
SEC1244 - Cops and Robbers: Simulating the Adversary to Test Your Splunk Security Analytics	Hunting / IR
SEC1547 - Splunk Security Essentials: What's New and What's Awesome	Hunting / IR
SEC1538 - Security Ninjutsu Part Five: The Most Advanced Content Money Can Buy	Hunting / IR
FN1209 - Visualize This, Mother Trucker	Visualizations
FN1398 - Splunk and the Machine Learning Toolkit in Action: Customer Use Cases	Data Science
SEC1979 - Splunk Phantom at Starbucks	Orchestration
SEC1898 - Pour Oil Not Sand Into Your Security Operations Center	Orchestration
SEC1233 - Hacking Your SOEL: SOC Automation and Orchestration	Orchestration
FN1913 - Old Meets New: Syslog and Splunk Connect for Kafka	Kafka
FN1211 - Don't Miss the Bus -- Splunking Kafka at Scale	Kafka
FN1184 - Unleashing Data Ingestion from Apache Kafka	Kafka
SEC1905 - 159 Security Use Cases in Record Time with Splunk and Kafka	Kafka
FN1629 - Exciting, To-Be-Announced Platform Session	Roadmap
FN1508 - Exciting, To-Be-Announced Platform Session	Roadmap
SEC1987 - What's New in Splunk for Security	General Security
SEC1983 - Splunk User Behavior Analytics (UBA): Methods and Best Practices to Get Started Now	UBA
SEC1275 - Monitoring and Miigating Insider Threat with Splunk Enterprise and Splunk UBA	UBA
SEC1982 - Splunk UBA Tunes Down the Volume at Shentel	UBA
SEC1796 - Addressing Alert Fatigue and Threat Hunting with Analytic Stories	ES
SEC1310 - Enterprise Security Biology Revisited: Dissecting the Asset and Identity Frameworks	ES
SEC1570 - Enterprise Security Health Check	ES
SEC1479 - Say Goodbye to Your Big Alert Pipeline, and Say Hello To Your New Risk-Based Approach	ES and Risk



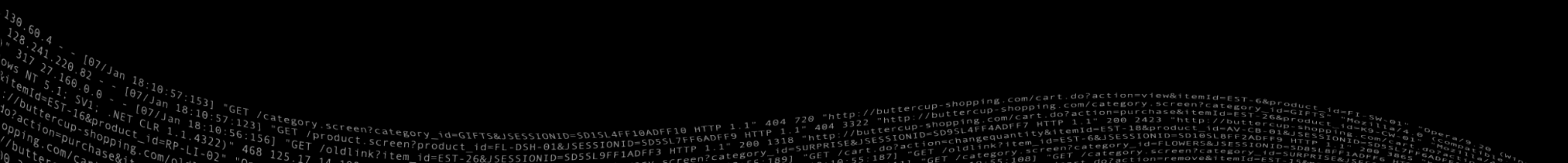
splunk> .conf18

Key Takeaways

Building advanced analytics is easy with a solid foundational dataset

The age where you needn't worry about low noise detections is past. If you don't have a solution for this, you're behind the curve.

If you find sessions and apps like this useful, please rate us in the app so that Splunk provides have more people build things like this.



Thank You

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