.conf2015

Security Jiujitsu Building Correlation Searches in Splunk

David Veuve

Senior SE, Security SME, Splunk

splunk>

Disclaimer

Personal Introduction

- **David Veuve** – Senior Sales Engineer for Major Accounts in Northern California
- Pre-Sales Security SME
- dveuve@splunk.com
- Former Splunk customer (for 3 years, 3.x through 4.3)
- Security guy
- Primary author of *Search Activity* app
- Search Ninja

Agenda

Four types of security correlation rules you probably want

1. Correlation across many sourcetypes and events
   – Generate high fidelity alerts
2. Privileged user monitoring
   – Detect lateral movement and prioritize risky users
3. Conquering alert fatigue
   – Don’t be beaten by tens of thousands of alerts per day
4. Threat Intel hits
   – Immediately detect bad actors in your environment

For Each Scenario

- Focus on Visibility, Analysis, and Action
- Driven by customer use cases and/or real deployments!
- Backed by A Splunk App!

Alternative Agenda

Sweet Searches

Who Are You?

1. Security Engineer / SOC Analyst / Threat Analyst / Someone Technical Who Cares about Security
2. Splunk skill level is basic-advanced
3. No Enterprise Security required (though it can make things easier at scale)

What Experience Are You About to Have?

- | eval state=If(SplunkExperience<Ninja, “Information Overload”, “Neato”)

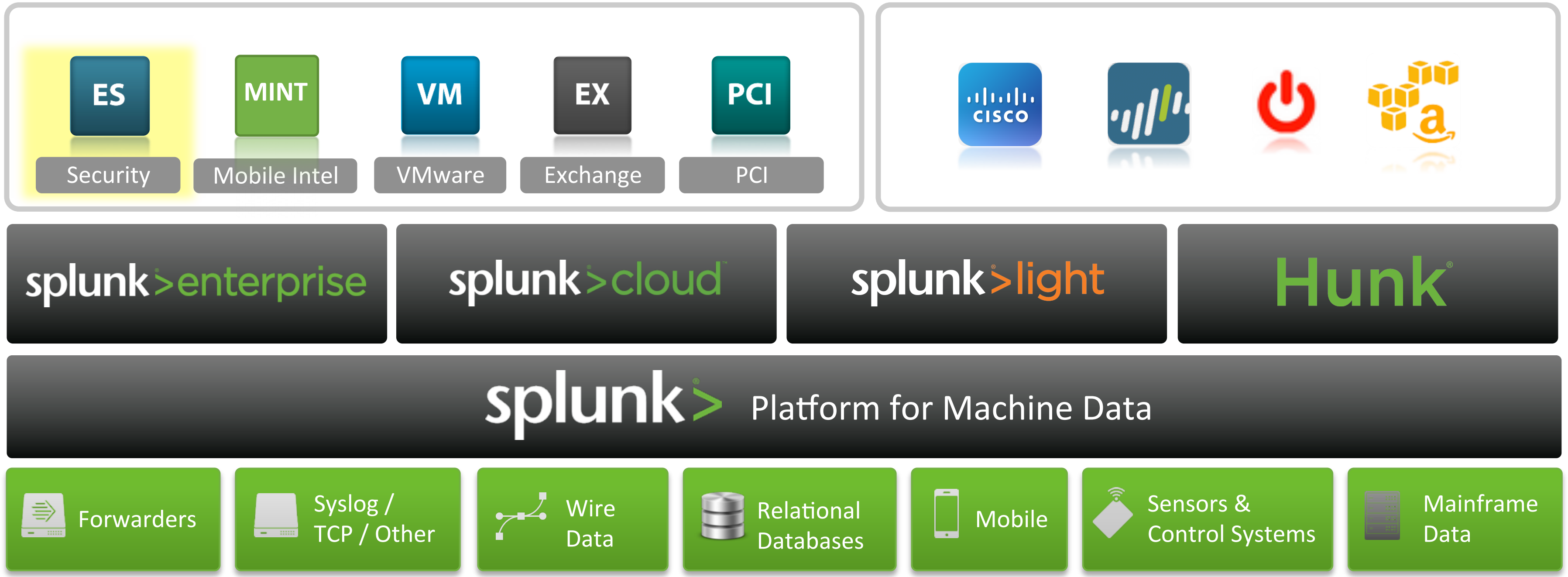
- Don’t fear – the app is (almost) here. Published at Session End.

- Guide to Search for Correlation Use Cases

Security Correlation In Splunk

Splunk Solutions > Easy to Adopt

Across data sources, use cases & consumption models



Splunk Correlation Rules

- Easy in enterprise security
- But even in core Splunk Enterprise, any search can:
  - Send an email
  - Trigger ServiceNow/Remedy/any other ticketing system
  - Run a script
  - Interact with other systems to block / increase logging for hosts
- Correlation in Splunk is just searching

Visibility – Analysis – Action

- Framework for evaluating data and responding Splunk
- Applies to all existing frameworks, as it's the Splunk side of the loop.
- For example, Let's look at the lateral movement section of the kill chain.
  (Not familiar with the kill chain? It's a great way to understand the phases of an attack. Check the URL below.)
- Visibility: What data will let you detect Lateral Movement?
- Analysis: What will you do to that data to come to a decision?
- Action: What will you do in response to that decision?
  – Can we automate all of this?

Kill Chain: http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf

Visibility, Analysis, AND Action

Correlation Across Multiple Sourcetypes

Correlation Across Multiple Sourcetypes

- Some don't know how to correlate across multiple sourcetypes in Splunk
- Easy: Across many auth log types
- Easy: Across auth logs and event logs
- Easy: New process launch within 15 minutes of anti-virus event also with uncategorized proxy activity
- In the following slides, we'll see several techniques useful when creating correlation searches, followed by one complete example.

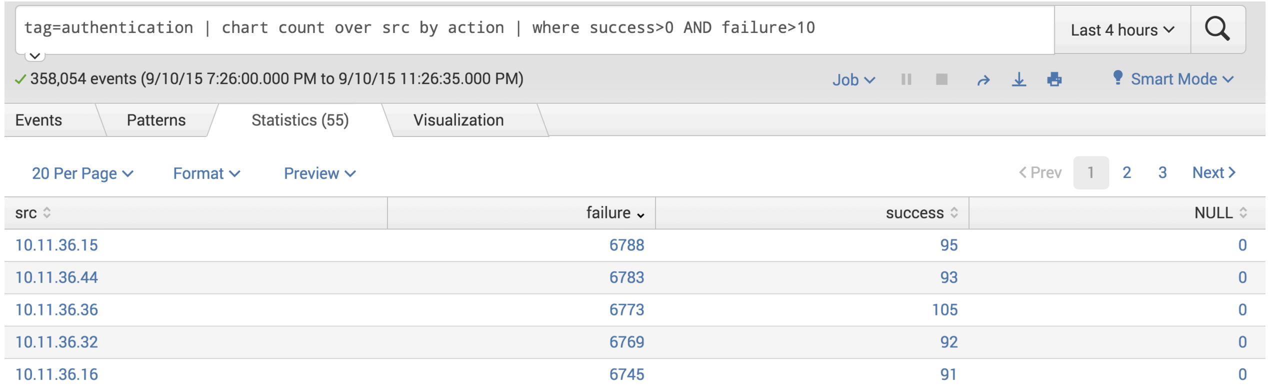
Techniques – Common Information Model

```
tag=authentication | chart count over src by action | where success>0 AND failure>10
```

- If you leverage Splunk's Common Information Model you can write one search across many products.
- The above search could cover twenty different products, all with matching field extractions

- Most searches in this session will be based on the common information model
- Try with the ES Sandbox!

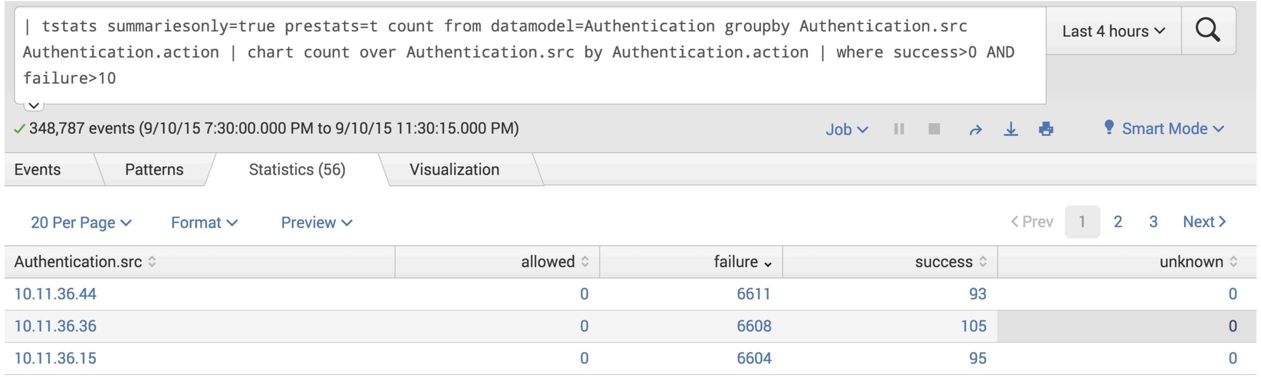
Search Example



```
tag=authentication | chart count over src by action | where success>0 AND failure>10
```

✓ 358,054 events (9/10/15 7:26:00.000 PM to 9/10/15 11:26:35.000 PM)

src	failure	success	NULL
10.11.36.15	6788	95	0
10.11.36.44	6783	93	0
10.11.36.36	6773	105	0
10.11.36.32	6769	92	0
10.11.36.16	6745	91	0



```
| tstats summariesonly=true prestats=t count from datamodel=Authentication groupby Authentication.src
Authentication.action | chart count over Authentication.src by Authentication.action | where success>0 AND
failure>10
```

✓ 348,787 events (9/10/15 7:30:00.000 PM to 9/10/15 11:30:15.000 PM)

Authentication.src	allowed	failure	success	unknown
10.11.36.44	0	6611	93	0
10.11.36.36	0	6608	105	0
10.11.36.15	0	6604	95	0

Raw Search

- 71 Seconds

With Data Model Acceleration

- 9.8 Seconds

Techniques – Flexible Stats

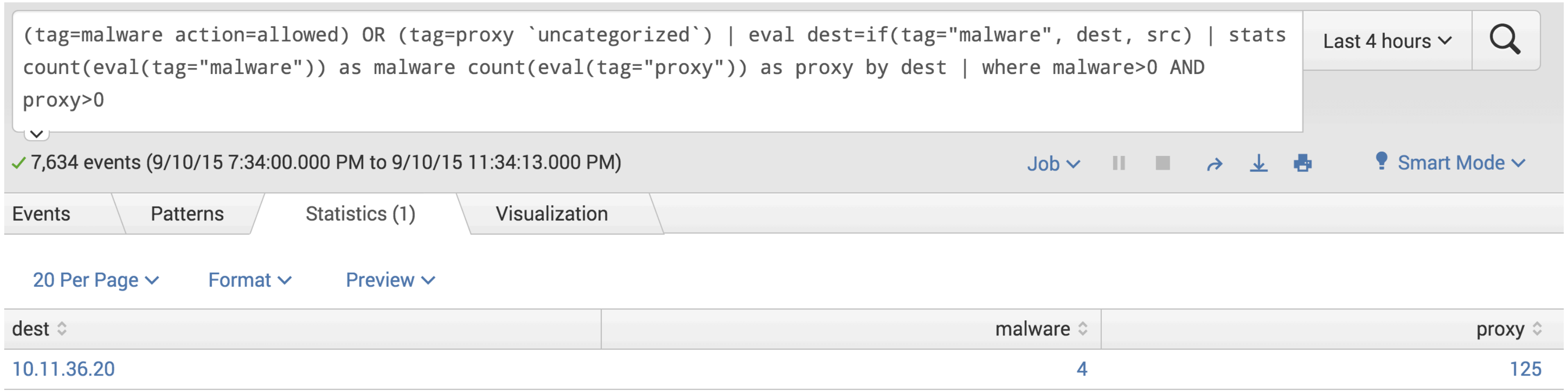
Example: | stats count(eval(action="success")) as successes count(eval(action="failure")) as failures by user

- Anything you can do inside of an eval you can do in stats
- Great Techniques:
  - If statements (use null for non-valid results)
    - values(eval(if(action="success",user,null))) as "Successful Users"
  - Searchmatch and match for flexible matching
    - 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
  - AND OR NOT
    - If(match(…) AND (host="emailserver" OR host="webserver")…)

Techniques – Expand Base Search

- 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

Search Example



```
(tag=malware action=allowed) OR (tag=proxy `uncategorized`) | eval dest=if(tag="malware", dest, src) | stats
count(eval(tag="malware")) as malware count(eval(tag="proxy")) as proxy by dest | where malware>0 AND
proxy>0
```

dest	malware	proxy
10.11.36.20	4	125

Techniques – The Other Stats

- Sometimes you need more flexibility
- Transaction is the world's biggest Swiss army knife, but usually you just want a screwdriver
- Consider:
  - streamstats – ordered processing
  - eventstats – additive (non-destructive) stats processing
  - geostats – be world aware
- If necessary: transaction

Techniques – Breaking Subsearch Limits

- Common Usage: [search index=malware | table host] index=proxy
- Interpreted as: (host=victim1 OR host=victim2) index=proxy
- Easy specificity creates huge performance improvements
- (Did you know you can do | eval myhost=[search tag=malware | return dest])
- Subsearches limited to 10,000 results and 60 seconds by default
- You can also return a literally interpreted search string:

[search tag=malware | stats values(dest) **as search** | eval search="(dest=" . mvjoin(search, " OR dest=") . ")"]

- Can't break 60 second limit without limits.conf change

Techniques – Higher Confidence

- Trigger your components and register to a summary index
  - Hey, ES does that already!
- Example: Find sources or destinations of brute force, victims of IDS hits, or malware events (clean or not) and determine if those hosts have new uncategorized web proxy activity
- We'll look at that later

Core Use Case

- New Process Launch within 15 minutes of anti-virus alert (successful or failed) and uncategorized proxy activity
- [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

First, find our infected hosts. (Didn't we just say use stats instead of subsearch? Well, it's a guideline, but with a 5 minute window, this should be reasonable.)

Pull endpoint + proxy data for those hosts

See how many proxy and endpoint events per host

Filter to just hosts that have the known bad events

How Does it Scale?

- Tstats version of Core Use Case
- | 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

Pull Malware Data

Pull Web (Proxy) Data

Pull Endpoint Data

Normalize Field Names

Do Count

Scalability Improvements

- Raw Search: 21 seconds
- Tstats without “summariesonly”: 2.76 seconds
- Tstats with “summariesonly”: 2 seconds
- Would likely have to remove summariesonly, due to implicit data model lag

About Endpoint Logs

- Curious about Endpoint Monitoring? Check out the epic talk from Splunk Rockstar James Brodsky:

Splunking The Endpoint

Wednesday, September 23, 2015 | 3:15 PM-4:00 PM

Demo

Privileged User Monitoring

Privileged User Monitoring

1. Start by detecting something bad
2. Focus on highly visible or highly privileged users.

Visibility: Authentication Data Model (OR auth events)

Analysis: stats + lookup + eval

Action: Creating notable with appropriate severity and risk

Visibility

- Authentication Data Model – could be any auth data you wish, but the auth data model is available to everyone (even if you don't have ES) and gives us a common language to speak

Analysis

- Alert when users who usually log into very few systems all of a sudden logs into a lot
- tag=authentication earliest=-30d@d| bucket _time span=1d | stats count by user, host, _time
- | eval isRecent=if(_time>relative_time(now(),"-1d"), "yes", "no")
- | stats avg(eval(if(isRecent="no",count,null))) as avg first(count) as recent by user, host
- | eventstats count(eval(if(avg>0,"yes",null))) as NumServersHistorically dc(eval(if(recent>0,"yes",null))) as NumServersRecently by user
- | eval Cause=if(isnull(avg) AND NumServersHistorically>0, "This is the first logon to this server", "")
- | eval Cause=if(NumServersHistorically*3 < NumServersRecently, mvappend(Cause,"Substantial increase in the number of servers logged on to"), Cause)
- | where Cause!=""

Pull daily count per host, per user per day

Identify recent logins

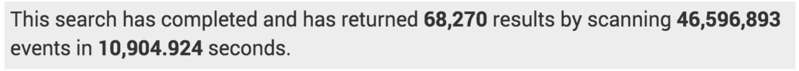
Pull average and recent per user, per host

Analyze cross-host context

First time / existing users

Track users who log in more servers than normal

Filter atypical users


This search has completed and has returned **68,270** results by scanning **46,596,893** events in **10,904.924** seconds.

Analysis

- Alert when users who usually log into very few systems all of a sudden log into a lot
- | tstats summariesonly=true count from datamodel=Authentication where earliest=-30d@d groupby Authentication.dest Authentication.user _time span=1d | rename Authentication.dest as dest Authentication.user as user
- | eval isRecent=if(_time>relative_time(now(),"-1d"), "yes", "no")
- | stats avg(eval(if(isRecent="no",count,null))) as avg first(count) as recent by user, dest
- | eventstats count(eval(if(avg>0,"yes",null))) as NumServersHistorically count(eval(if(recent>0,"yes",null))) as NumServersRecently by user
- | eval Cause=if(isnull(avg) AND NumServersHistorically!=0, "This is the first logon to this server", "")
- | eval Cause=if(NumServersRecently>3 AND NumServersHistorically*3<NumServersRecently,mvappend(Cause,"Substantial increase in the number of servers logged on to"), Cause)
- | where Cause!=""

Pull daily count per host, per user per day

Identify recent logins

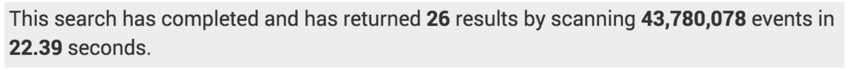
Pull average and recent per user, per host

Analyze cross-host context

First time / existing users

Track users who log in more servers than normal

Filter atypical users


This search has completed and has returned **26** results by scanning **43,780,078** events in **22.39** seconds.

Analysis – Part Two

- You know the high risk, high exposure users in your organization
  - Sys Admins
  - Executives
  - Contractors
  - First 3 months of employment, last 3 months of employment
- Sources:
  - AD Group Membership
  - AD Title
  - HRIS Employment Status
- 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

Analysis – Part Two - Example

```
| 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
```

Analysis – Putting it Together

```
[… insert your Privileged User Activity 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”)
```

Analysis – Part Three (ES Specific)

- If you are using Enterprise Security, you can pass a severity, risk_score, risk_object, and risk_object_type fields in your search
- This lets you override the severity, or risk associated with the system. | rename AggRisk as risk_score
- Better yet, use the built-in ES Identity Framework to automatically set the urgency for your events!

Demo

Conquering Alert Fatigue

Conquering Alert Fatigue

- There are more events than you can possibly handle. Fact of life.
- There are many great techniques for managing this:
  - Risk analysis to identify machines/users/etc of the greatest concern
  - Statistical analysis to identify unusual hits
  - Track alerts on multiple vectors to bubble up significant threats
  - Trigger increased logging after mundane alerts (Action!)
  - Machine Learning to identify outliers
- We will look at each of these techniques

Visibility

- A typical tier one analyst can handle one event every 10 (maybe 15) minutes on average. That’s about 50 events per shift.
- If you have 10 tier one analysts, you shouldn’t have more than 500 events per day.
- For any rule triggering more than a hundred events per day, consider these techniques. Certainly rules with tens of thousands.

Analysis Technique – Risk-Based

- This approach is great for general purpose events, and should always be enabled
- Increase the risk associated with an entity (user, system, signature, etc.) for each time that it hits, and then focus activity on high risk entities
- Available out of the box with Enterprise Security (index=risk_activity)
- Easy to implement on your own by adding | summaryindex index=risk entities of your own

Analysis Technique – Statistical – Part One

- This technique requires some experimentation, and threat modeling. **You need to know your environment**, to know what you want to learn about.
- Establish your base dataset:

```
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
```

Pull IDS Data

Check Daily Destinations per Signature

Pull relevant metrics per signature, per host

Pull relevant metrics per signature overall

Analysis Technique – Statistical – Part Two

Inclusive: Potentially Leaves you Exposed

| fields severity signature dest avg stdev earliest recent_count avg_earliest avg_num_per_dest sig_wide_count sig_wide_recent_count

(Remind us of the fields)

| lookup AssetPriority as host OUTPUTNEW priority

Augment with any more context!

| where (avg_earliest > relative_time(now(), "-1d")) OR (earliest > relative_time(now(), "-1d") AND (recent_count / sig_wide_recent_count > 0.1 OR priority>3)) .....

Include what you want to be alerted on, aggregated!

Analysis Technique – Statistical – Part Two

Exclusive: Potentially more work

```
| fields severity signature dest avg stdev earliest recent_count avg_earliest avg_num_per_dest sig_wide_count sig_wide_recent_count

| lookup AssetPriority as host OUTPUTNEW priority

| where NOT (avg_earliest < relative_time(now(), "-1y" AND sig_wide_recent_count / sig_wide_recent_count < 0.05 AND priority <=3)
.....
```

Repeat of last slide

Exclude what you don't want to be alerted on!

Analysis Technique – Combine Multiple Vectors

- If you have a number of correlation searches firing, you can track the output of those searches and do a meta analysis.
- If you use Enterprise Security, use index=notable. If you use a ticketing system, query that. If you use the alert manager, use | rest "/services/alerts/fired_alerts"
- Craft a search that looks for multiple event endpoint alerts, and then create a high confidence high severity event based on that.

Analysis Technique – Combine Multiple Vectors

- Example:

```
index=notable | stats dc(search_name) as NumRules
values(search_name) by dest| where NumRules>2
```

- More Powerful Example:

```
(index=notable Antivirus OR ids) OR (index=proxy
category=uncategorized) | eval dest=case(index="proxy", src,
index="notable", dest) | stats count(search_name) as NumRuleHits
count(eval(index="proxy")) as NumUncategorizedHits by dest
```

- In ES >= 3.2, search index=risk_activity for correlations w/o notables

Analysis Technique – 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. With or without ES, you can use your ETDR solution. Many customers leverage panblock 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.

Analysis Techniques – 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.
  - See Booth in the Solutions Showcase
  - See Sessions Wed 10 AM
  - Ask your SE!
- Look at the new ML App!
  - See booth in the Solutions Showcase
  - See Sessions Wed 2:15 PM, Wed 4:15 PM (Also, AWS Wed 12:15 PM)
  - Ask your SE! (Watch him look bewildered!)

Demo

Threat Feeds

Threat Feeds

- With everything covered above, you could easily build your own threat feed engine in Splunk.
- **Don’t**

Great Threat Feed Tools

- Enterprise Security is officially supported and will integrate nine different types of threat intel

IPs	Process Names	Certificate Common Names
Domains	Hashs	Email Addresses
User Names	Certificate Hashes	File Names

- If you don't have Enterprise Security, look at Splice on Splunkbase – it's not supported, but works for many customers.
- Please, please don't build it yourself!

Demo

But That's Not all for Threat Intel

- Lots of things you can do with Threat Intel
  - Turning Indications of Compromise into Tangible Protection – Wed 3:15 PM Kaiser's Katie Winslow and Michael Slavick
  - Operationalizing Data Science Output Using Splunk - Wed 4:15 PM Kaiser's Dave Dyer and Tim Neyman
  - Managed Threat Intelligence in Splunk ES – Thurs 11:15 AM Splunk's Brian Luger (ES Developer)
- Generate it yourself (go ask your SE and tell him David Veuve sent you)

Wrap Up

We Looked at a Lot of Things

- Many search techniques
- Many correlation search examples

How to Be Successful

- Go download the app – install it on a non-production instance! Includes:
  - Link to session PDF
  - Link to hand-out
  - An example search for every use case and technique discussed!
- Go check out other sessions
- Post questions on answers.splunk.com with tag "correlationsearch"!
- Talk to the person next to you! Go attend *Birds of a Feather* Session

Give Me Feedback!

- Rate it in the app
- Also give feedback here: http://www.davidveuve.com/conf2015
- $50 Amazon gift card will be randomly given!

- Download the app, play around with it, and give me feedback http://www.davidveuve.com/conf2015app
- Another $50 Amazon gift card will be randomly given!

THANK YOU

Appendix
Network Hunting

Techniques - 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

Techniques - Frequency

- Understand your common ratios is easy – HTTP GET/POST/Connect/Delete
- 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

Techniques – Temporality

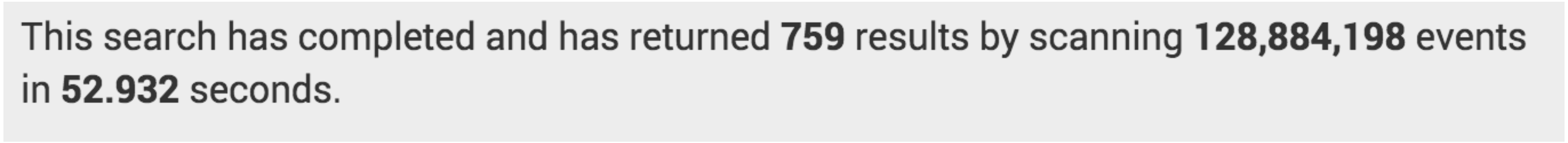
- Long URLs typically immediately follow short urls (or are to advertising servers)
- Examples:
  - https://goo.gl/maps/yjXdP
  - https://www.google.com/maps/place/250+Brannan+St[202 characters clipped]
- Detect with: streamstats
- Many activities occur only during 9-5, 8-6, or etc.
- Detect with: date_hour (if not global) or eval's strftime()

Techniques - Coherence

- Coherence (in this case) – Systems that are servers tend to stay servers, systems that are clients tend to stay clients
- Also useful for looking at network traffic

Analysis

```
| 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.

Analysis

```
| 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)
```