splunk> .conf2017

APT Splunking

Searching for adversaries with quadrants (and other methods)

David Doyle | CIRT Analyst, Bechtel

Andrew Hunt | Malware and Threat Intelligence Team Lead, Bechtel

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.

Who Are We?

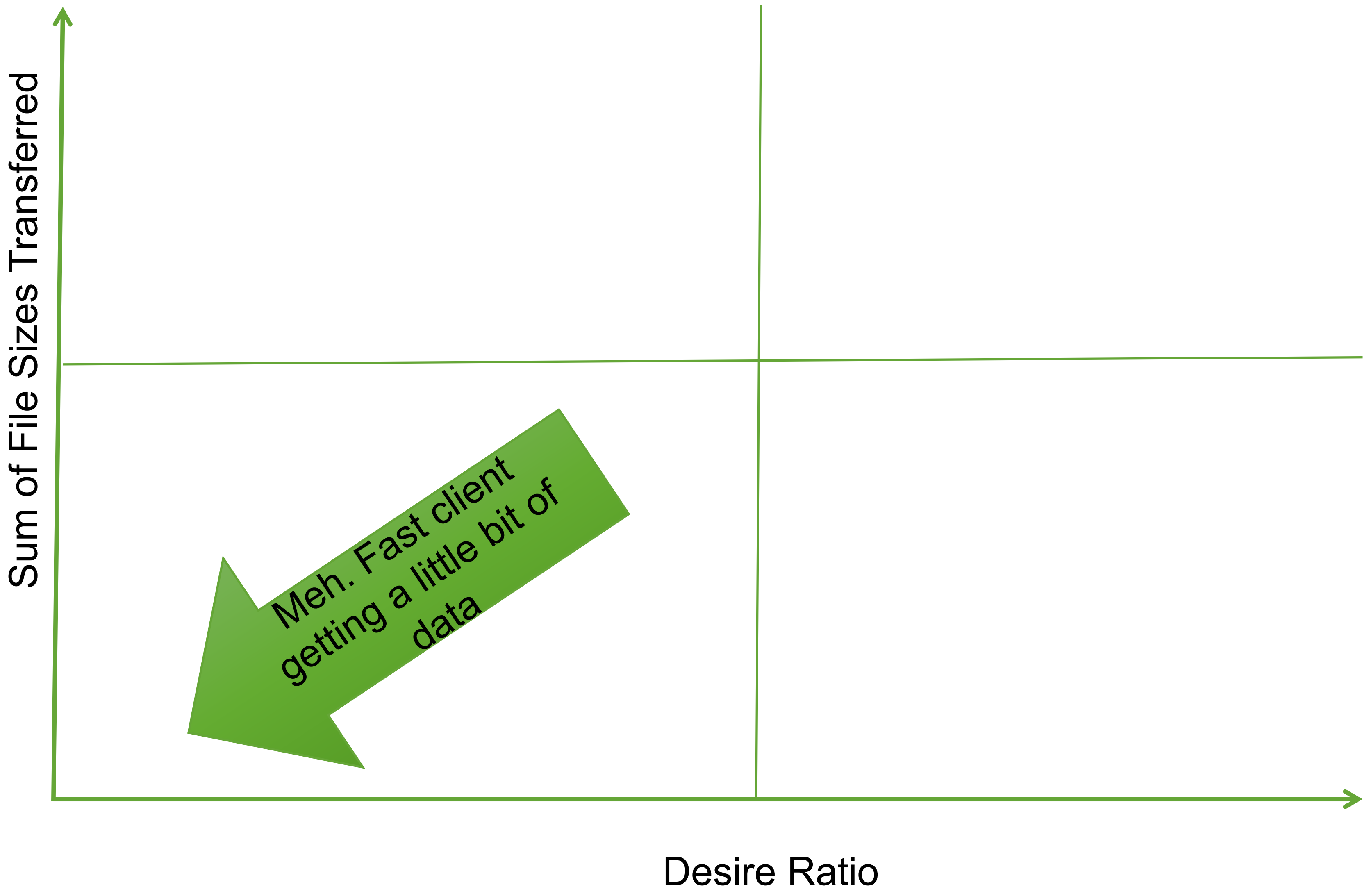
And why should you care?

- David Doyle
  - Bechtel CIRT Analyst
    - Splunk Administration
    - Viz Building
    - Incident Response
    - Plugging Visibility Gaps
    - Making it Look Easy
- Andrew Hunt
  - Bechtel Malware & Threat Intel Team Lead
    - Behavior analytics
    - Threat Intelligence
    - Math
    - Malware Analysis
    - IoT / DCS

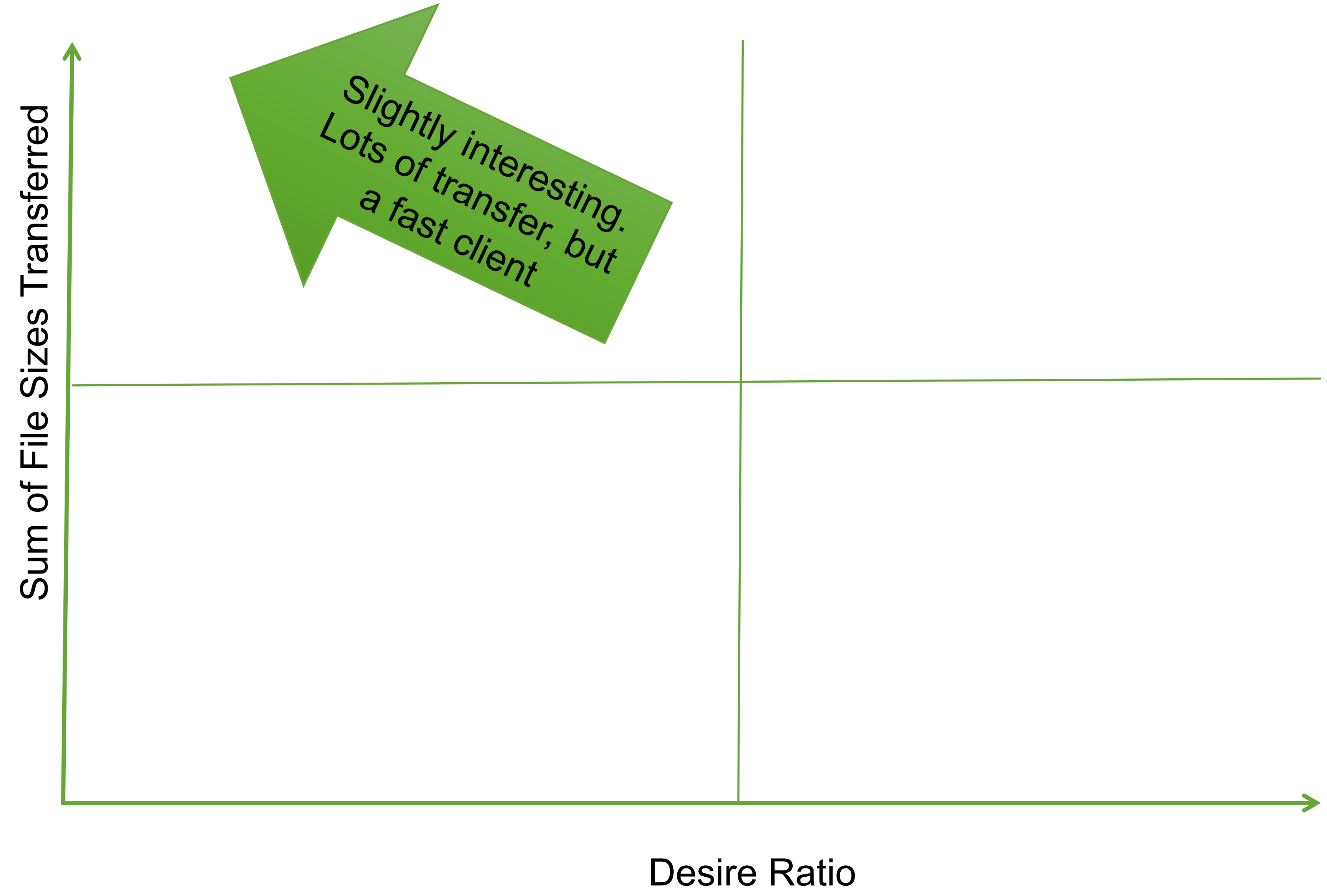
Quadrant Analysis for Dummies

OR, a brief reintroduction to that thing you already know

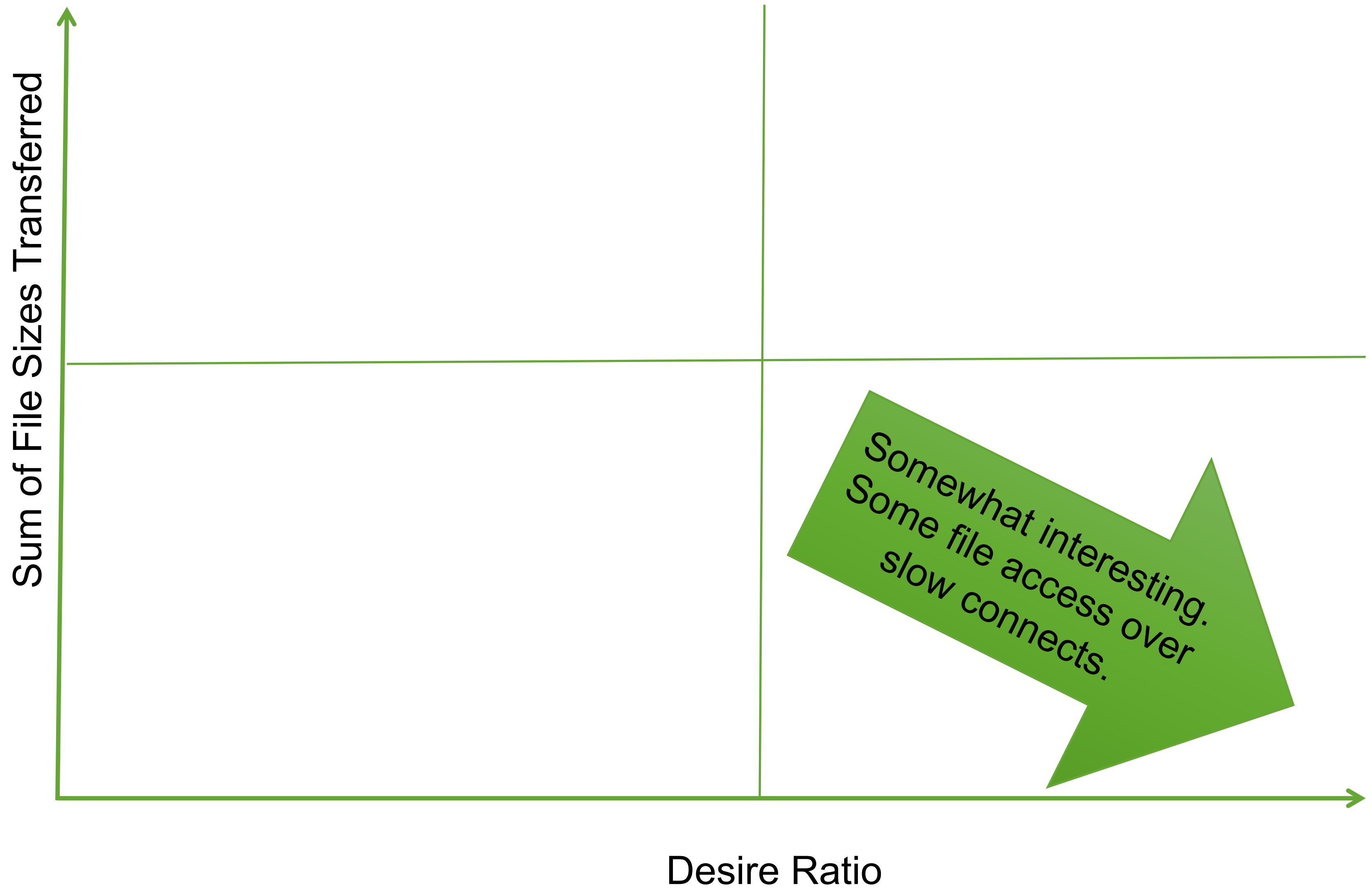
How Quadrant Analysis Works



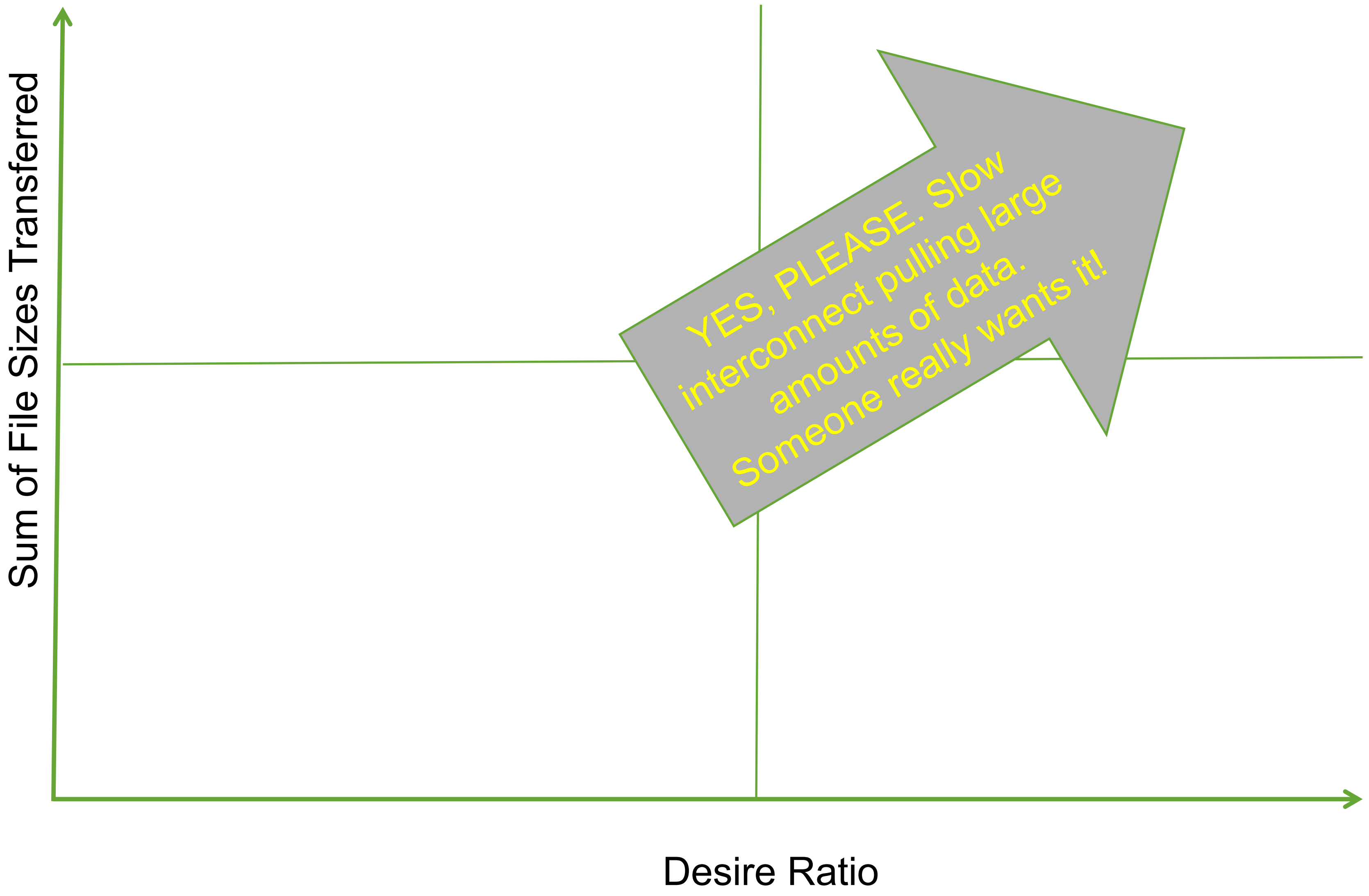
How Quadrant Analysis Works



How Quadrant Analysis Works



How Quadrant Analysis Works



Using Quadrants to Winnow the Field of Knowns

OR, Knowing What You Know

Use Case 1: Dicing Quadrants

- FTP server exfiltration pivot
- Application logs contain artifacts about file transfers
  - File size
  - File transfer time
  - Connected IP address
- Provides introspection on several features
  - Overall size transferred to each client (scale)
  - Velocity can be calculated for each connection (speed)
    - velocity = file size / transfer time

Cracking Addicts With Speed

- Why do we care about transaction velocity?
  - We can make hypotheses based on assumptions
- Assumed
  - Faster clients are closer
  - Faster clients are more legit when pulling large amounts of data
  - Aggressors will tunnel, which introduces latency, thus a slower session
  - Aggressors are geographically far away, which increases the time cost of the interconnect
  - Aggressors want to pull lots of data
  - Aggressors are not Bechtel IPs (RFC 1918, 147.1/16)

Reading the Tea Leaves

- Hypothesis based on previous assumptions
  - Some bad actors can be identified by their velocity characteristics
  - Clients that have fast transfer velocity are less suspicious
  - Clients that have slower transfer velocity are more suspicious
  - Clients that have slow transfer velocity that pull large amounts of data are highly suspicious

It’s a Quadrant!

- Four mathematical assumptions
- Two gradients (size vs. velocity)
- Linked by client address….

It’s a Quadrant!

- Four mathematical assumptions
- Two gradients (size vs. velocity)
- Linked by client address….

We can graph that!

Stalking Desire

- Invert velocity to create a ‘desire ratio’
  - desire_ratio = 1 / velocity
  - Should provide a value between 0 and 1
  - Low numbers indicate low desire
    - high velocity, low effort
  - High numbers indicate high desire
    - low velocity, high effort

Stalking Desire

- Invert velocity to create a 'desire ratio'
  - desire_ratio = 1 / velocity
  - Should provide a value between 0 and 1
  - Low numbers indicate low desire
    - high velocity, low effort
  - High numbers indicate high desire
    - low velocity, high effort

That's an understandable value!

Some Quick Adjustments

- Some transfers were reported as extremely slow
  - In the sub bps
  - Infinitesimal rate blew out the scale on desire ratio.
    - Probably an error
    - Can’t transfer in sub-bytes
  - Made adjustments to present a reasonable scale to analyze the rest of the data
    - 0.0001 < desire_ratio < 1
    - Beyond 1 is an error
    - Below 0.0001 is just too small to care about

Some Quick Adjustments

- There are a lot of small transfers
  - Clutter the bottom of the graph
  - Drag the filesize scale out of analyzable range
- Assumed that we are interested in transfer greater than 5MB
  - filesize > 5000000 bytes

The Query

```
host=LOC* index=ftp_from_host
| spath
| rex field=cliconnaddr "^(?<cliconnaddr_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):"
| rex field=lstnconnaddr "^(?<lstnconnaddr_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):"
| eval transfer_rate=filesize/transtime
| fillnull value='-' filesize transtime
| search NOT (cliconnaddr_ip=0.0.0.0/8 OR cliconnaddr_ip=0.0.0.0/12 OR
[more of your internal networks here, you get the idea]
OR transtime='-' OR filesize='-')
| eval desire_ratio=1/transfer_rate
| stats avg(desire_ratio) as a_desire_ratio, sum(filesize) as s_filesize by
cliconnaddr_ip
| where s_filesize>5000000 AND a_desire_ratio>.00001 AND a_desire_ratio<1
```

The Breakdown

```
server=ash* index=ftp_from_host
| spath
| rex field=cliconnaddr "^(?<cliconnaddr_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}):"
```

- Spath
  - Splunk's built-in structured path parser
  - Automatically parses and extracts self-tagged key-value data
  - XML, JSON
- If spath is the Easy Button, why the REX?
  - cliconnaddr includes the IP address AND the port
  - Multiple sessions probably have different ports
  - Want to link by IP address only, not port
  - Need to extract the IP address from cliconnaddr

The Breakdown

```
| eval transfer_rate=filesize/transtime
| fillnull value='-' filesize transtime
| search NOT (cliconnaddr_ip=0.0.0.0/8 OR cliconnaddr_ip=0.0.0.0/12 OR
[more of your internal networks here, you get the idea]
OR transtime='-' OR filesize='-')
```

- Transfer rate is calculated with ‘eval’
- Eliminate useless events
  - Irrelevant events that report to this host/index combo
  - Don’t have file size or transfer data, but screw with calculated results
  - ‘fillnull’ followed by ‘search NOT’ filters these events out of the data set
  - Also get rid of IP ranges assumed not to be suspicious

The Breakdown

```
| eval desire_ratio=1/transfer_rate
| stats avg(desire_ratio) as a_desire_ratio, sum(filesize) as s_filesize by cliconnaddr_ip
```

- Calculate the ‘desire ratio’ as the inverse of velocity
  - Codifies the hypothesis that pulling data over slow connections means you want it more
- Calculate the average desire ratio and sum of data transferred by the client IP address
  - Averaging the desire ratio smooths bumps that might occur over time
  - Summing file sizes provides a measure over the query time horizon, aggregating the time dispersion of low-and-slow data pulls

The Breakdown

| where s_filesize>5000000 AND a_desire_ratio>.00001 AND a_desire_ratio<1

- Implement filters to apply whitelisting assumptions
  - Only care about data transfers in excess of 5MB
    - YMMV. Adjust as needed.
  - Only care about a ‘desire ratio’ between 1/100000 and 1
    - It seems stupid, but it cuts a lot of high-bandwidth, legit transfers from the graph

But Are We Ever Going To Plot It?

- Run the query
- Engage the visualization engine
  - Format as a scatter plot
  - Adjust X- and Y- axes to logarithmic scales

But Are We Ever Going To Plot It?

- Run the query
- Engage the visualization engine
  - Format as a scatter plot
  - Adjust X- and Y- axes to logarithmic scales

AND NOW…

Shotgunning: Scatter Plot

- Lots of noise
  - But, separate!
  - And expected!
  - And all but filtered out on its own!

Shotgunning: Scatter Plot

- Up + Right = Interesting
  - X axis: desire_ratio
    - Rightward = slower
  - Y axis: filesize
    - Upward = larger
- So, up and to the right = slow and determined

Psychoanalysis Session

- Hovering over an interesting dot tells you the IP address
- Check out some quick features
  - DNS resolution
  - WHOIS
  - AS netblock ownership
  - Quick search for malice
- Does it smell bad?

What Did We Accomplish?

- Based on available data, math, and assumptions about demonstrated behavior
- Provided a method to filter down the amount of client IP addresses that need to be analyzed as a cold-call
- But of course…

What Did We Accomplish?

- Based on available data, math, and assumptions about demonstrated behavior
- Provided a method to filter down the amount of client IP addresses that need to be analyzed as a cold-call
- But of course…

Hard indicators always win out!

Quadrant Analysis on Undefined Traffic Data

OR, The “or other methods” part

Use Case 2: Undiscovered Country

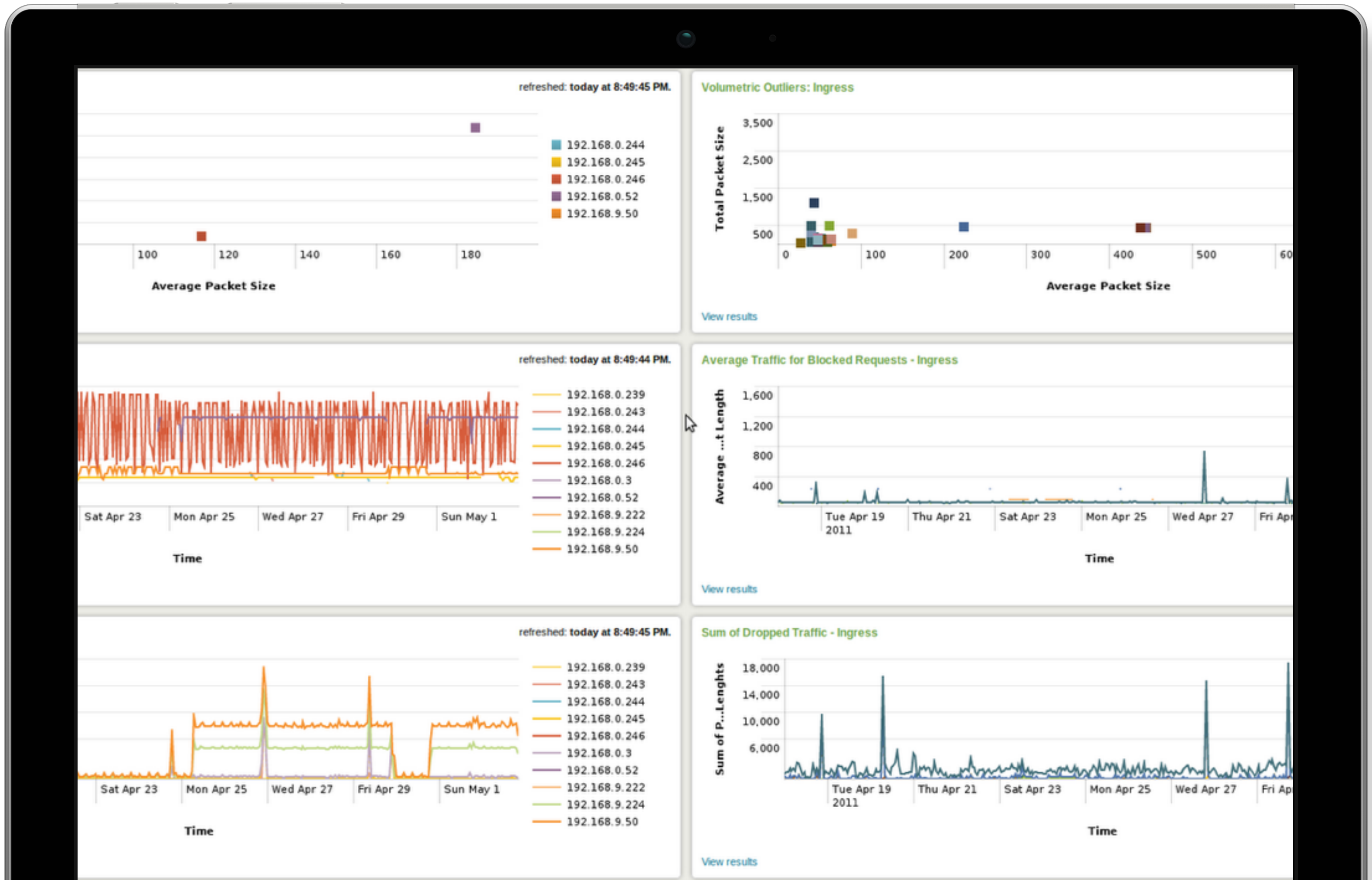
- Looked at artifacts from logs for a known activity
  - Discovery had already occurred
- What can we find with Quadrant Graphing on large, unknown datasets?

Use Case 2: Undiscovered Country

- Looked at artifacts from logs for a known activity
  - Discovery had already occurred
- What can we find with Quadrant Graphing on large, unknown datasets?

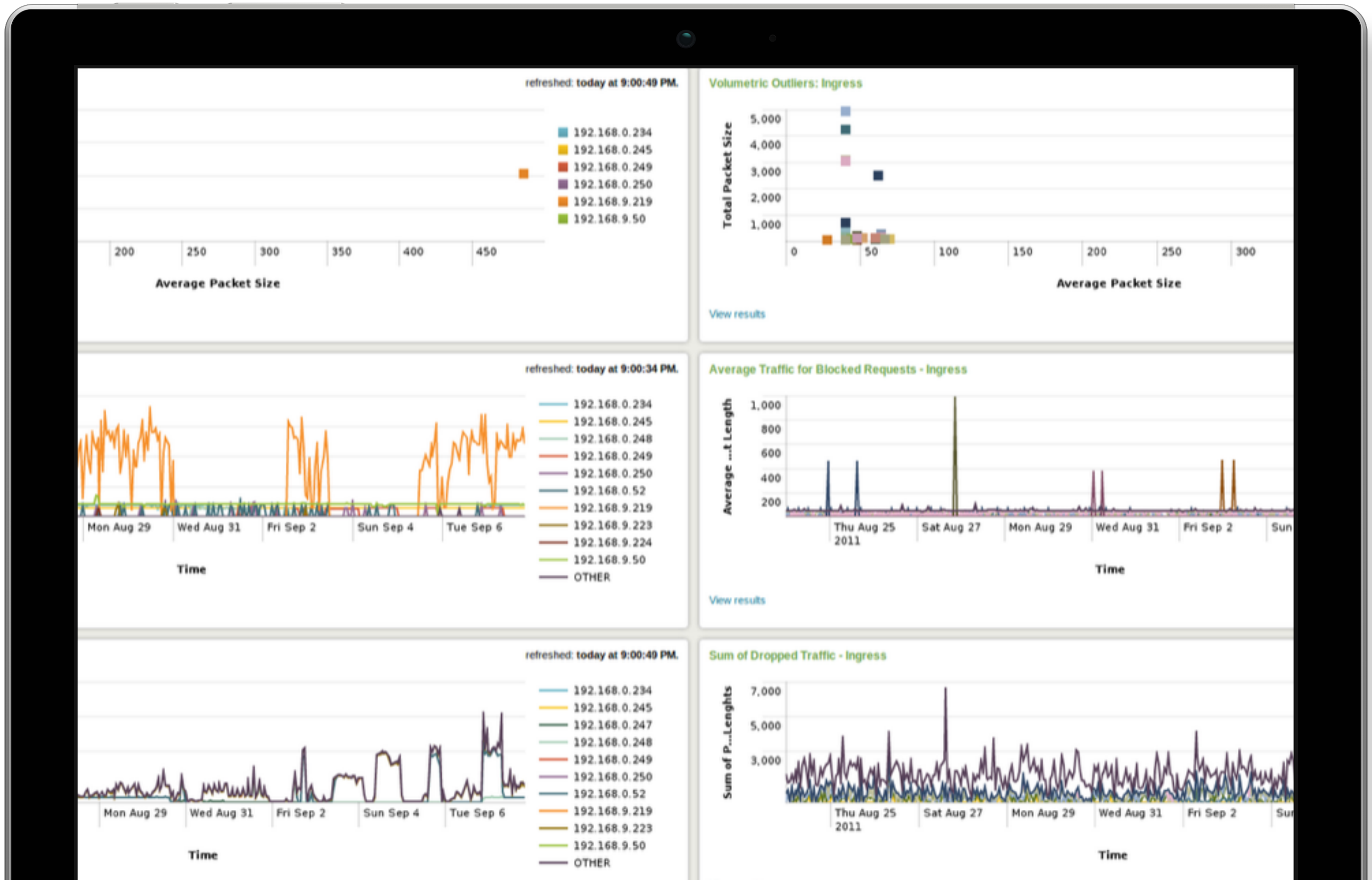
I'm so glad you asked!

Typical Heads-Up Dashboard



- These are all normal
  - …or at least expected
  - Don't worry about it

The Next Day…



- Things look very different, don't they?
  - Averages are normal, but steady, higher than expected baseline
  - Summation of dropped packets much higher than "normal"
  - Scatter plot shows several hosts w/small transactions

What's Different?

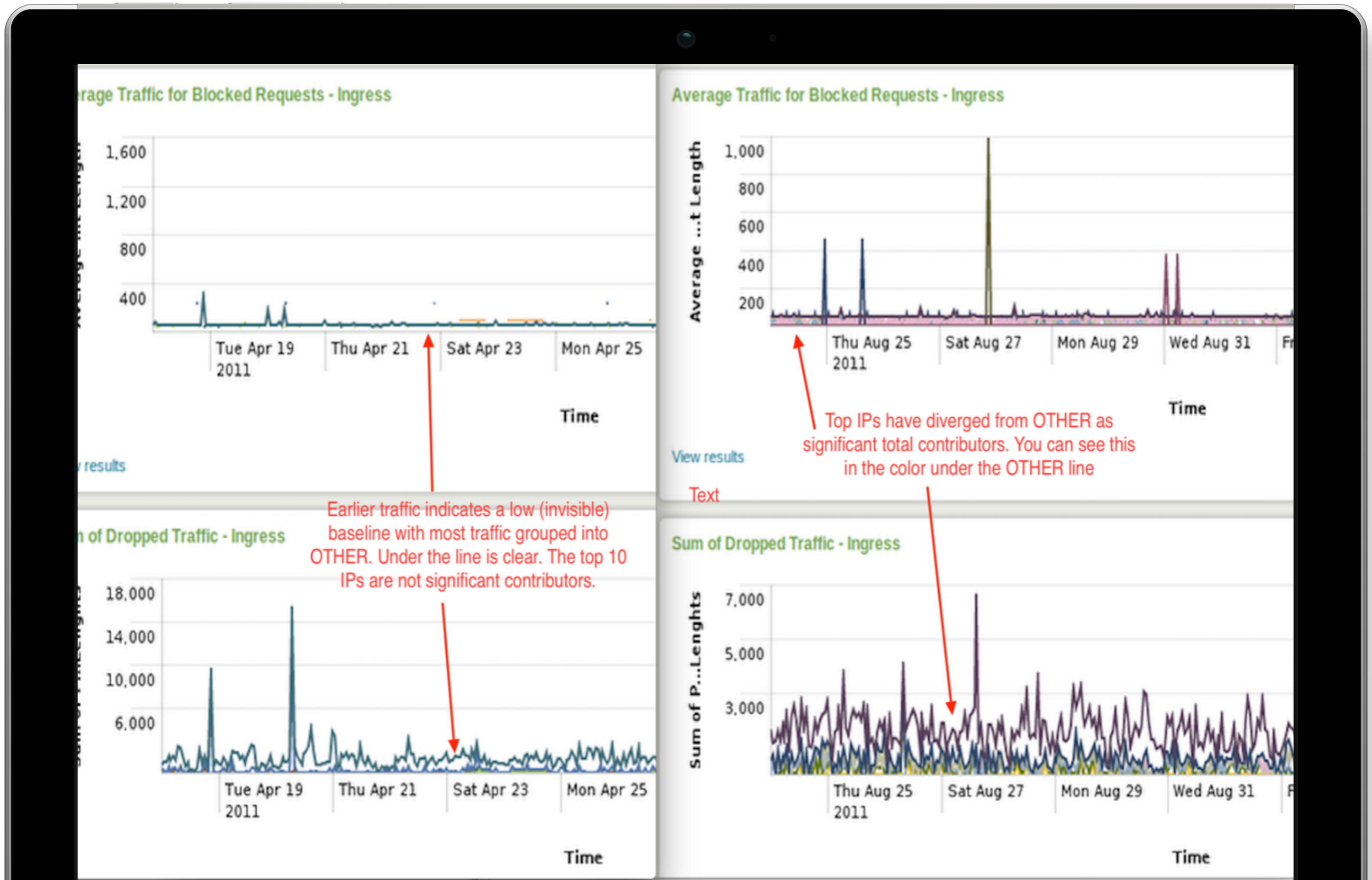
- Latter graph is more active, 'noisier'
- Ingress has less diverse drops
- Average vs Summation of packets reveals a clean ratio in the top 10

What's Different?

- Latter graph is more active, 'noisier'
- Ingress has less diverse drops
- Average vs Summation of packets reveals a clean ratio in the top 10

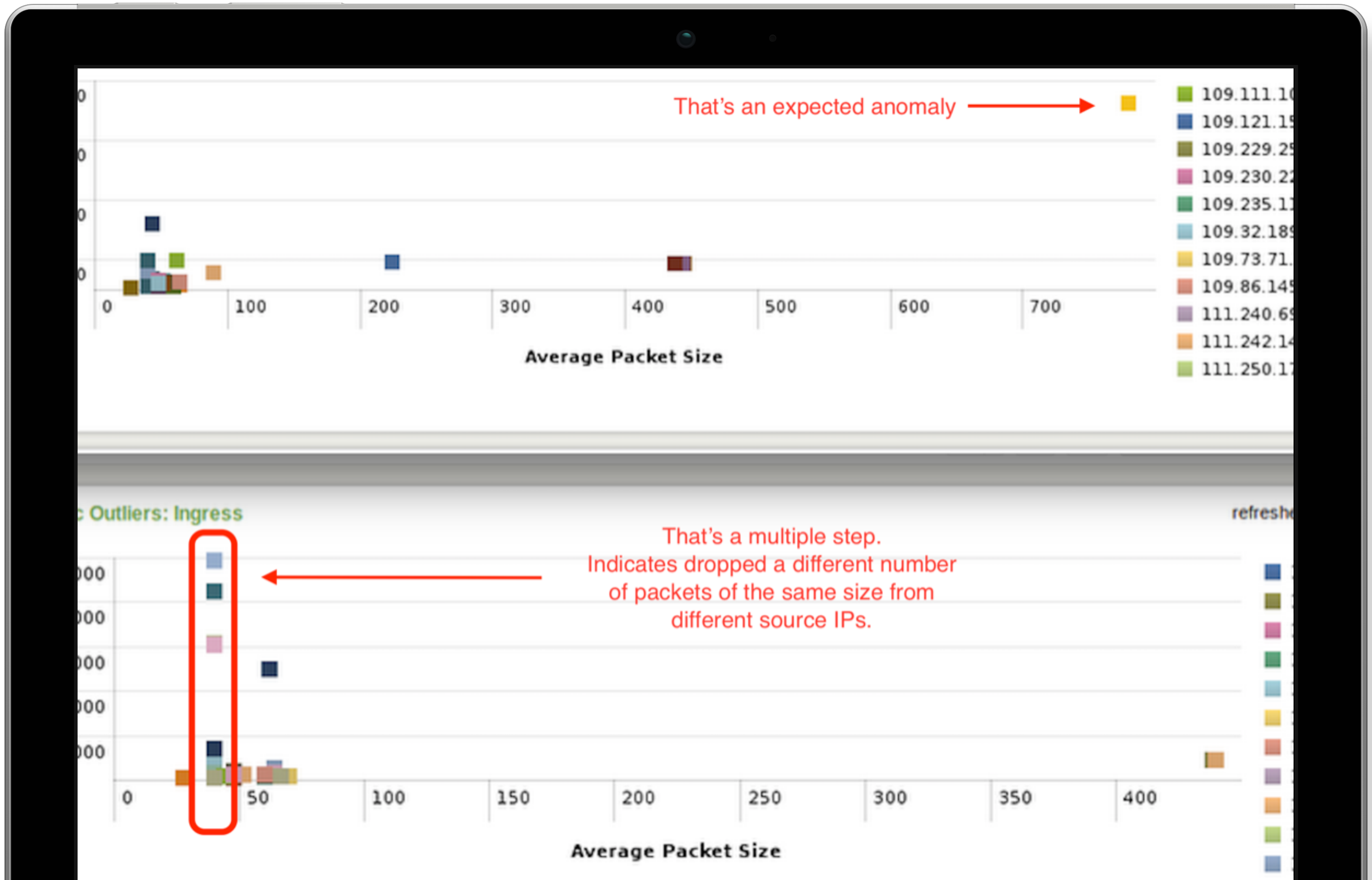
That's weird!

Less Diverse Drops



- Spikes in average blocked requests
- Lots of noise in sum of dropped traffic
- But, what is it?

The Stars Align



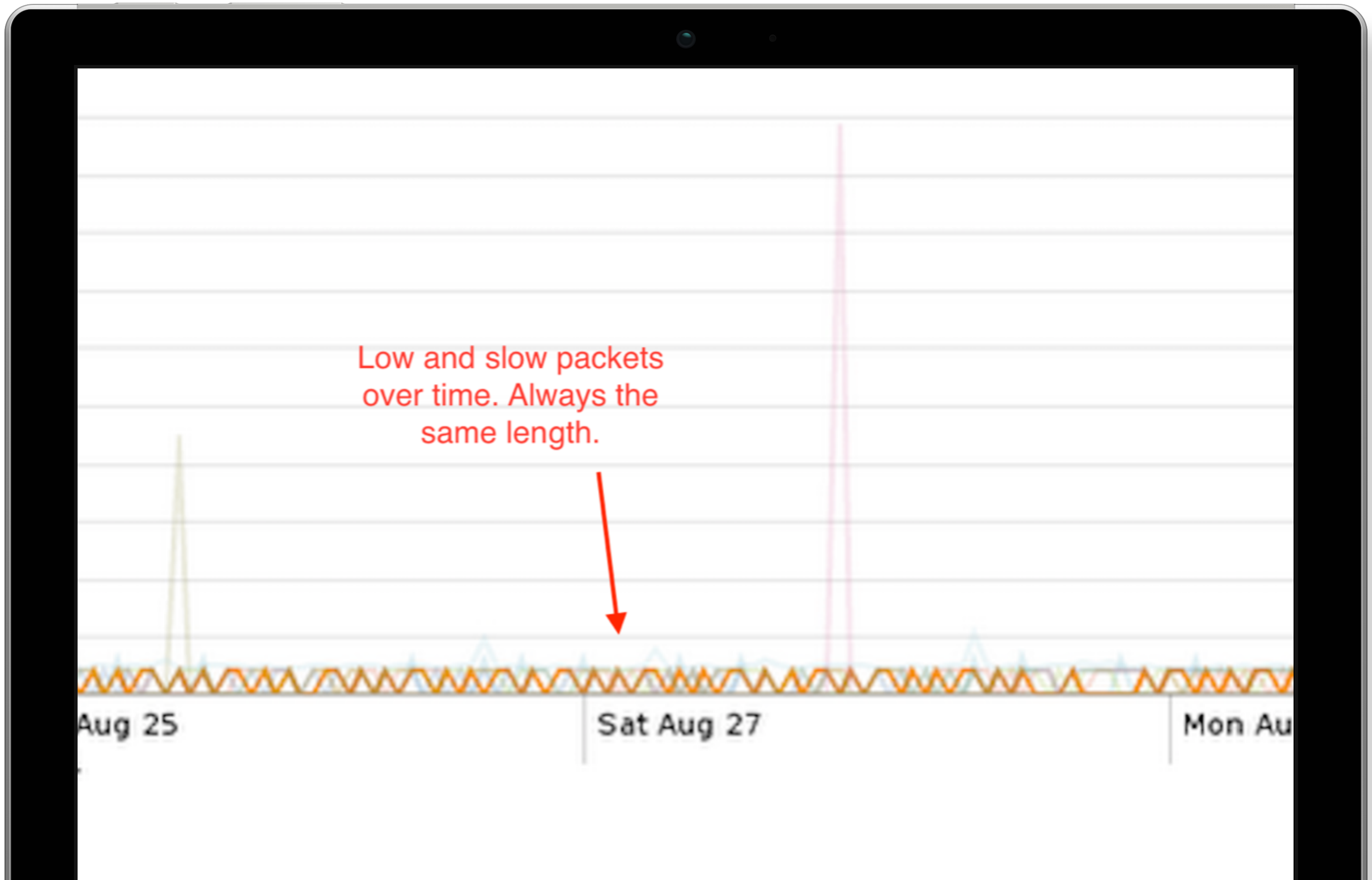
- Those pesky quadrants again
  - Nothing really jumps out for yesterday
  - But today's another story - what's up with that column?

So, Here's Another Query

```
sourcetype=firewall decision=b AND NOT
(s_ip=192.168.* OR s_ip=169.254.1.* OR s_port=80 OR s_port=443)
| fields s_ip, pkt_len
| timechart limit=0 span=1h sum(pkt_len) by s_ip
```

- Looking at firewall drops
  - Filter out local address space and known garbage
  - Filter out huge streams like HTTP[s] that tunnel everything. Not enough context at this level of analysis
- Limit to just the fields you want. Speeds search. The optimizer can reduce the number of fields it needs to parse.
- Sum packet lengths by source IP and display in a visual timechart

Shh! I'm Hunting Wabbits….



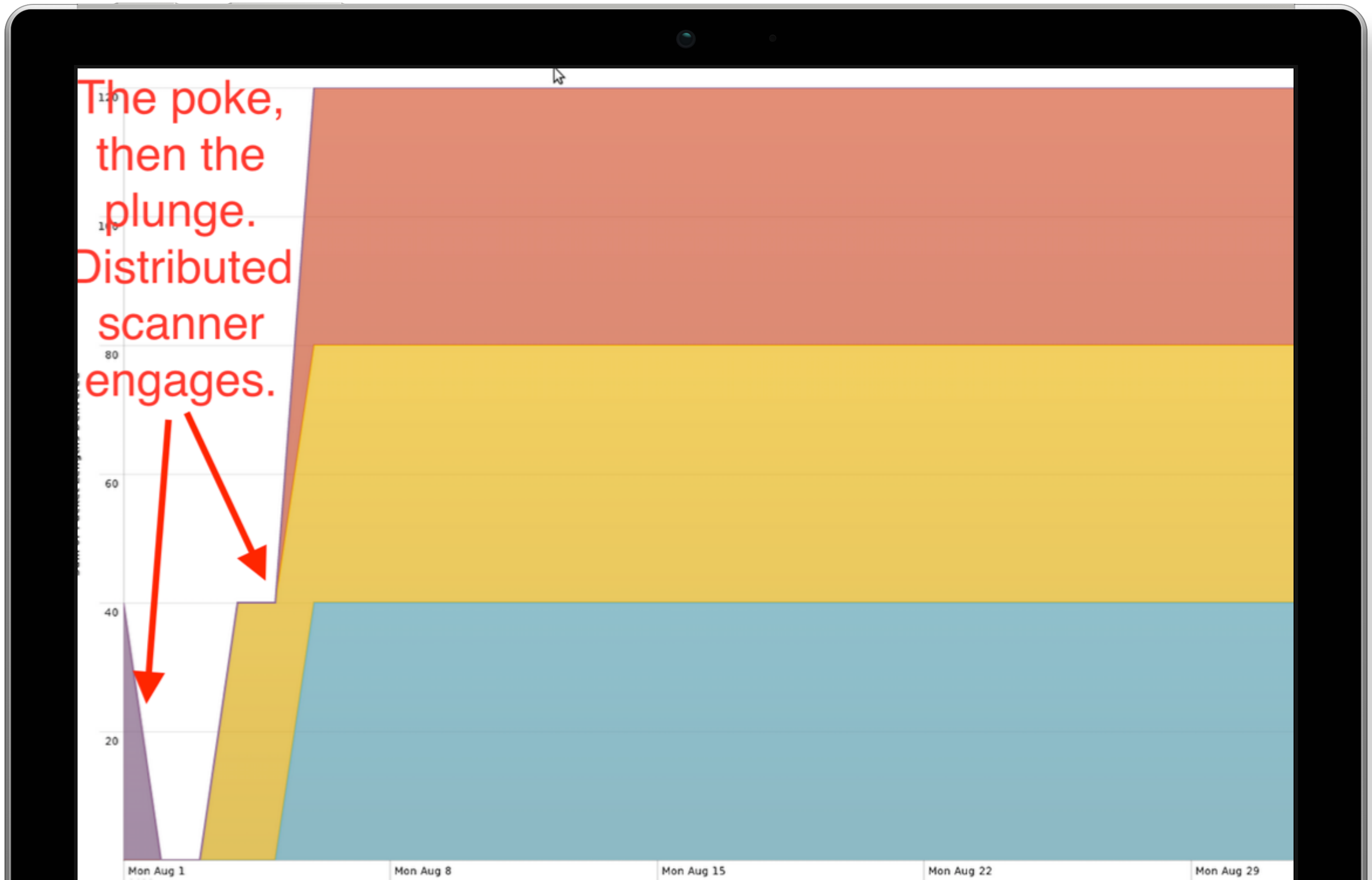
- Further hunting
  - Treat null values as 0
- One block ended up showing low and slow activity
- Time to investigate further….

More Hunting Means More Queries

```
sourcetype=firewall decision=b 58.218.199
| fields s_ip, pkt_len
| timechart limit=0 span=1d avg(pkt_len) by s_ip
```

- Search firewall events for specific network
  - Full text indexer parses on punctuation and spacing
  - Designed for IP addresses and domains!
  - CIDR field match notation also available
- Limit fields
- Chart in time by the average packet length for the subnet

Wabbit Season, Meet Duck Season



- Stacked area chart
  - Treat null values as 0 (again)
- Distribution even between scanners
  - Single host used as a preliminary sniffer

Fudging Fudd

- We have discovered a distributed scanner
- Have a fair idea of some of the infrastructure
- What is it looking for?
- Intent?

Fudging Fudd

- We have discovered a distributed scanner
- Have a fair idea of some of the infrastructure
- What is it looking for?
- Intent?

What hunting season is it?

Wow, Much Query, Very Splunk

```
sourcetype=firewall decision=b 58.218.199
| fields s_ip, d_port
| dedup s_ip, d_port
| stats count as source_scanners by d_port
| sort -source_scanners
| lookup portServices port as d_port OUTPUT service as service
| table d_port, service, source_scanners
```

- Search prior subnet with field filters for s_ip, d_port
- Dedup source IP/dest port since only interested in counting the number of services hit
- Count and sort. This orders the numbers for the visualization
- We built a quick lookup table. You can, too!

It's Open-Proxy Season!

	d_port	service	source_scanners
1	1080	socks	3
2	2301	compaqdiag	3
3	2479	ssm-els	3
4	27977	TDSS open proxy	3
5	3246	Kademlia P2P (mlnet)	3
6	5390	Bosch Video Management System Software-open proxy	3
7	6588	AnalogX-proxy	3
8	7212	GhostSurf open proxy	3
9	73	netrjs-3	3
10	80	www	3
11	8000	iRDMI	3
12	8008	http-alt	3
13	8080	http-alt	3
14	8085	OKOGATE open proxies	3
15	8088	omniorb	3
16	8090	http-alt-alt	3
17	8118	privoxy	3
18	8123	polipo proxy	3
19	9000	CSlistener	3

- Proxys upon proxies
  - Upon proxies
- Each one shows up with three scanners
  - Look familiar?

Playing Favorites?

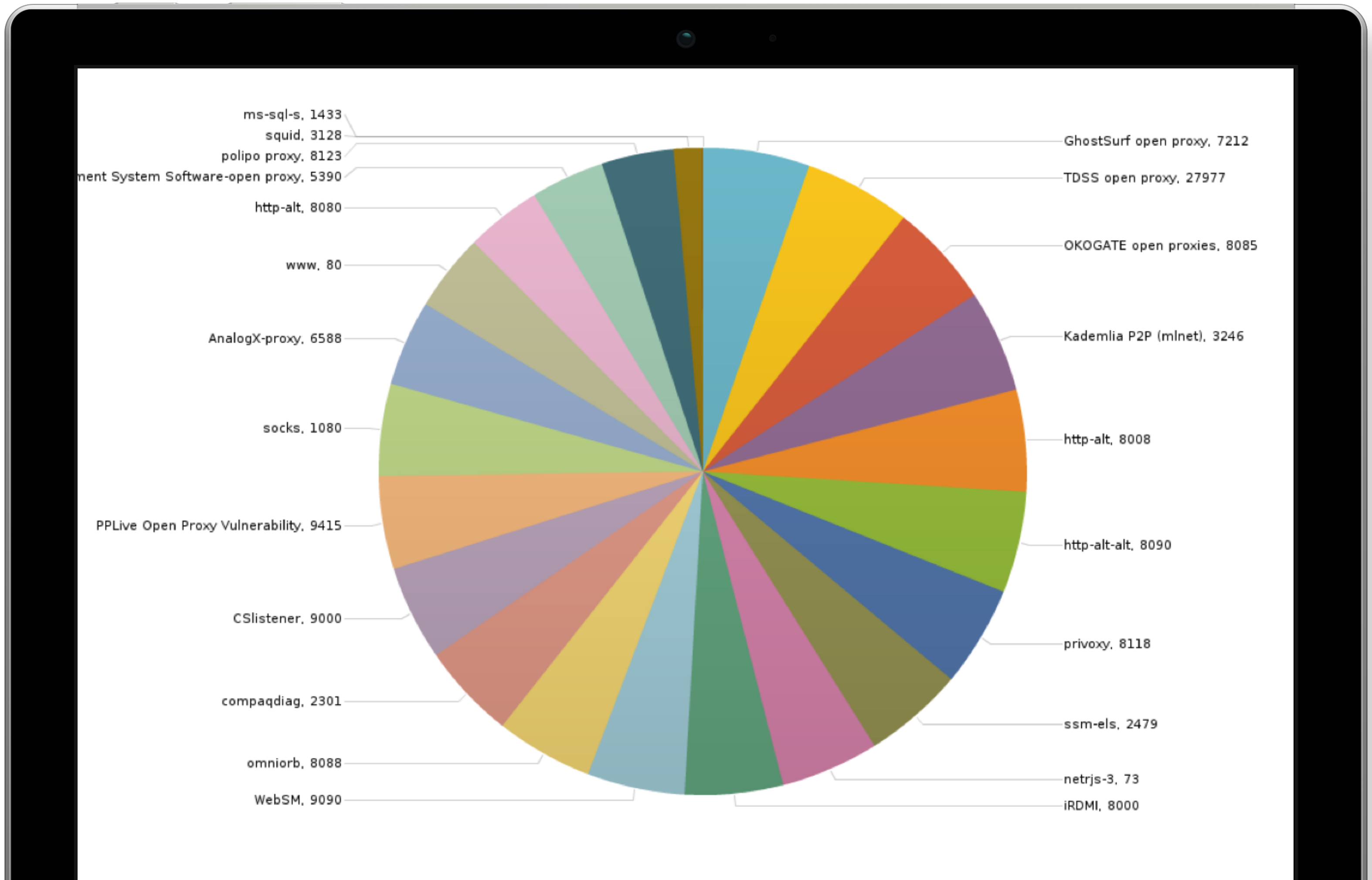
- Scanners appear to hit each service in the individual node runs
- REVERSE PERSPECTIVE
  - Sometimes this reveals other anomalies
  - Does the cluster favor certain services?
  - Does it look for one thing more than the others?

Revising That Last One

```
sourcetype=firewall decision=b 58.218.199
| fields s_ip, d_port
| lookup portServices port as d_port OUTPUT service as service
| strcat service ", " d_port label
| stats count as source_scanners by label
| sort -source_scanners
```

- Same base search
- Augment data with port descriptions
- Concatenate the text data into a label
  - Pie chart only accepts a text column and a number
- Count the hits
- Sort to make visualization nicer

EEO Compliant Proxy Hunter



- Pie chart, names and ports
- Hope you're not colorblind
  - (David is. Don't ask him to count wedges.)

What Did We Learn?

- Found a distributed scanner
- Linked the scanning nodes simply by packet size, time proximity, and math
- Looking for open proxies from poorly configured services and leftover malware
- Scanner is pretty static. Same packets
- Scanner looks evenly for proxy ports, no favoritism

Thank You

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

splunk> .conf2017