Office 365 in Nearly That Many Days

MS Cloud logging, then and now

David Doyle | Bechtel

October 2018

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

PART 0: The Obligatory Intro Section

DAVID DOYLE

Splunk Puncher, Bechtel

splunk> .conf18

Managing Expectations

What this is about – and what it isn't

What it isn't:

- A Microsoft presentation
  - I'm not a Microsoft employee
- A Splunk presentation
  - I'm not a Splunk employee
- A sales presentation
  - Unless you want to buy a dam or something, and even then, I'm probably not authorized to sell company services

What it is:

- A post-mortem
  - What we've done, what we haven't done, how it's worked out
- A cheerleading session
  - This stuff can be done, I promise
- A therapy session
  - Hoooooo boy…

Seven Core Takeaways

The tl;dr slide

- You can do it on your own
  - Apis exist, documentation exists, people who can do the job (theoretically) exist
- But IT'S A MASSIVE PAIN, and SOMEONE HAS DONE THE LEGWORK
  - Apps, tas, unofficial but proven methods, etc.
- Make a plan, and engage your stakeholders
  - This means EVERYONE
- Know your infrastructure
  - Your own network, your company's policies, MS idiosyncrasies, etc.
- Don't be afraid of the big scary data
  - It's data, you use data
- ????
  - Secret mystery point
- Know your use cases
  - Big, complex systems = big, complex possibilities

PART I: Past History

What we used to do, and why we stopped

Prehistory: What We Had to Work With

- A man, a plan, a canal: MS Cloud!
  - Around 2015 (before my time, before Splunk TAs went live)
    - 1x infrastructure engineer:
      - proficient in python and familiar with scripting inputs
    - 1x universal forwarder
      - sitting in DMZ, ready and waiting for such things
    - 1x institutional directive:
      - On High: “Get O365 logs in”

Prehistory: What He Built

- Lovingly handcrafted python harvesting module
  - Authentication, token management, log enumeration, log harvesting, checkpoint management, throttling/rate limiting, error/warn condition handling, etc.
- NOT a scripted or modular input in the traditional sense
  - Operated completely separate from Splunk
    - (Not out of necessity, it just happened that way)
  - Logs brought into localdir, simple inputs.conf stanza to send to Splunk

Prehistory: Did It Work?

- Yes.
  - With caveats
    - Which we'll deal with later
  - Auth: functional
  - Log harvesting: functional
  - Checkpointing: functional
  - Rate limiting/throttling: functional
  - Error handling: functional

Point One: You Can Do It On Your Own

- One person, sufficiently motivated and educated, can build it from scratch
  - Handle all basic important functions of log harvesting
    - Auth, logging, checkpointing, etc.
- Can be done to your spec, in your environment, by your rules, under your purview, tweaked and maintained as you see fit
  - Respond to issues on your schedule
  - Add new features on your schedule

Going It Alone: Worth It?

Pros vs. Cons

Advantages

- Get things that prebuilt solutions don't get
  - If prebuilt solutions don't exist, >0% is better than 0%
- Make changes on your own time
  - React to API, schema, etc. changes as fast as you want

Disadvantages

- Requires considerable onboard knowledge
  - O365 APIs
  - Harvesting methods
- Breakages are your problem
  - Nobody else coded this, so nobody else is fixing it
- Documentation is…kinda meh
  - Some things are unclear, others undocumented
- Critically, all the advantages are drying up
  - As solutions get better, reasons for not going to them get worse

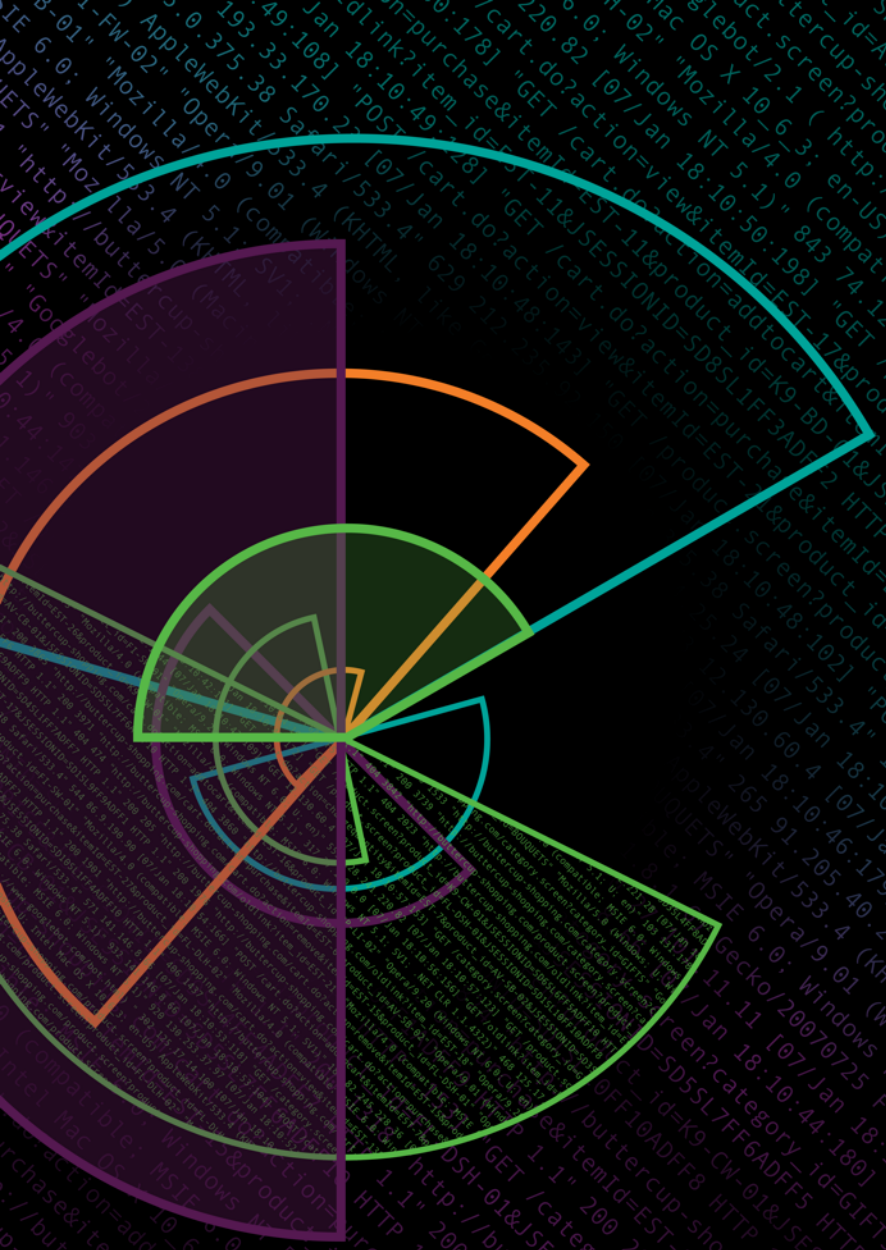
Point Two: But It's a Massive Pain, and Someone Has Done The Legwork For You

- Wizards aren't that common
  - SMEs in multiple areas usually have multiple responsibilities
  - Spells that can be performed by apprentices can be easier and cheaper to build out and maintain
- Splunk knows which way the wind is blowing
  - And weirdly, MS kind of does, too
- Fairly large TA environment for log harvesting from a variety of MS Cloud sources
  - Office 365
  - Azure
  - Subgroups of both

But, Do They Work?

Time to find out…

splunk> .conf18



PART II: Present…um, history

How we do it now

More History

Switching from handcrafted inputs to commodity sources

- Occasionally, sorcerers have other things to do
  - So their responsibilities get passed off to sorcerer's apprentices
    - And the apprentices ain't got time to learn all that
    - And in the meantime, vendors have put some work into it

More More History

The long, slow trickle

- Started with Office 365, using the Splunk Add-on for MS Cloud Services
  - Tl;dr: Implementation worked
  - Ryan Lait's Splunking Office 365 Data blog series helped
- But, further down the line, "Office 365 logs" kept ending up being "missing"
  - This is a lie, because I am perfect and run according to spec in 100% of cases
    - The real answer? The "missing" logs weren't Office 365 logs
    - But On High doesn't know that, they just know that YOU said it was working and it CLEARLY IS NOT
- So, how do you do it better?

Step One: Engage Your Stakeholders

- Your group
  - Decides what's important in your day-to-day
  - Your supervisor probably wants to know if this is going to take a minute
- Branch out to your org
  - The rest of IT: May have input on a tactical/operational/strategic level
  - C-suite: They decide who gets paid
- And yes, even vendors
  - Even MS devs
    - I know, I know….

But Why Deal With Other People?

- You aren't a one-person shop
  - If you're the Splunk person, someone in your org knows more about O365 than you
  - If you're the O365 person, someone knows more about Splunk than you
  - Either way, someone will probably know more about the org's needs and technical reqs than you
- …unless you are a one-person shop
  - In which case, you should be doing this presentation, not me
  - And while you're here, who's running the shop back home?

Point Three: Make a Plan, and Engage Your Stakeholders

- Figure out what you need to do FIRST
  - Read the docs
  - Learn about the TAs
  - Listen to the .Conf talks
    - *HELLO FROM THE PAST, FUTURE TRAVELERS*
- Create a start-to-finish plan that has input from stakeholders
  - CYA, CYA, CYA

Step Two: Know Your Environment

Internal networks

- Your internal network structure will affect how you connect Splunk to cloud services
  - DMZ? Firewalls?
    - Who owns those?
    - What's the RFC procedure for poking holes for weird ports like 5671?
    - Or URLs or groups of them?
  - ExpressRoutes?
    - Do you use them?
    - Who's in charge of them?
    - Do they know what they're doing?
  - Ingestion endpoint
    - Where will it live? DMZ? Internal network? *The cloud?*

Step Two: Know Your Environment

MS Cloud

- Your MS Cloud posture will determine what TAs you need
- Unfortunately, this gets confusing:
  - Using Office 365? Then the Splunk TA for MS Cloud Services
    - Eventually, Splunk TA for O365 (when it gets stable)
    - Except for Azure Active Directory, you'll want the Microsoft Azure Active Directory Reporting Add-on for Splunk for that
      - Except some event types DO come in through the MS Cloud Services TA
  - Using Azure? Then you'll want the Azure Monitor Add-on for Splunk
    - Unless you want to try out Azure Function for Splunk
    - Or pushing data into an Azure Storage Blob and pulling it via the MS Cloud Services TA
      - But beware, the data you get that way will be changing Nov. 1
- Confused yet?

Point Four: Know Your Infrastructure

- This can hang you up and cause delays
  - Especially if you have onerous or unwieldy RFC processes
  - Or project or service owners aren't known
  - Or if groups are trying to push back against movement that has already been approved
  - Or [INSERT YOUR FAVORITE DELAY HERE]

Step Three: Execute

Plan, implement, curse, wail, gnash teeth, GOTO 10 (Office 365)

- Phase I: Splunk Add-on for MS Cloud Services
  - Do this first (after planning phases)
    - Longest TTL, highest ROI
    - Lengthy but effective tutorial available from Splunk
    - Many of the things you build in AAD here will be used elsewhere
- Phase II: Supplemental TAs
  - Microsoft Azure Active Directory Reporting Add-on for Splunk
    - AAD events – you should have these even if you “aren’t” using Azure
  - Microsoft Office 365 Reporting Add-on for Splunk
    - Email message trace events – do you use Exchange Online? If yes, get this working

Step Three: Execute

Plan, implement, curse, wail, gnash teeth, GOTO 10 (Azure)

- Phase III: Azure Monitor Add-on for Splunk
  - Another high TTL, high ROI TA
  - Azure activity events, diagnostic events, metrics events
  - Easily the most complex install procedure
    - Multiple dependencies not included in package
    - AMQP transport mechanism for logs
- Phase IV: Supplemental TAs
  - Microsoft Azure Inventory Addon for Splunk
    - Azure inventory data

Step Three: Execute

Things we skipped

- Azure data via Splunk TA for MS Cloud Services
  - Works, but log format method for logs harvested this way is changing per MS
    - Also, TA-Azure_Monitor is working for us
- Azure Log Integration
  - Will be deprecated June 2019
- Azure Functions for Event Hub data
  - Replicates TA-Azure_Monitor data, and I didn't feel like building out an HEC

PART III: Understanding and Visualizing Data

The part you probably came for

splunk> .conf18

Let's Talk About Data

- Lots and lots of data available
  - You may not be familiar with it
    - If not, how do you integrate it?
- It's not rocket surgery
  - Just because it's out in the cloud doesn't mean it's fundamentally different
  - What is Office 365?
    - An application suite that lives somewhere else
  - What is Azure?
    - An infrastructure…suite, I guess you call it…that lives somewhere else

So, What's The Difference?

- The data came from somewhere else
  - Considerations for availability
    - It'll take longer to get to you
    - Outages for services can affect log availability, too
- The data looks Microsoft-ey
  - JSON, which I love, but YMMV
  - Microsoft-specific schemas which are both not CIM and less efficient than CIM
    - Because why say src_ip when you can say ClientIpAddress instead?
    - They don't even capitalize the P in ClientIpAddress, what the hell
- But that's really it, IMO, and you've been dealing with things like this already, so…

Point Five: Don't Be Afraid Of The Big, Scary Cloud Data

- It's not any more special than any other mission-critical application and infrastructure data from a source that your organization doesn't have total control over that nonetheless has a high dollar value attached to it
  - Just take that in for a second. Yeah.
- You deal with these all the time
  - You just might have to deal with this one more
    - it replaces some traditional on-prem solutions, supplements others

But What Do You Do With All That Data?

- …I dunno.
  - Only kinda joking here.
  - Managing Expectations, Pt. II: every implementation is a special snowflake
- What do YOU need from the data?
  - Do you use Office 365? Azure? Both? How much of each? In what capacity?

Point Six (Is Just Point Three Again)

- To refresh your memory: ”Make a plan, and engage your stakeholders”
- If you had a game plan and you engaged your stakeholders, you would know the answers to this
  - So if you didn’t listen, GO DO THIS NOW

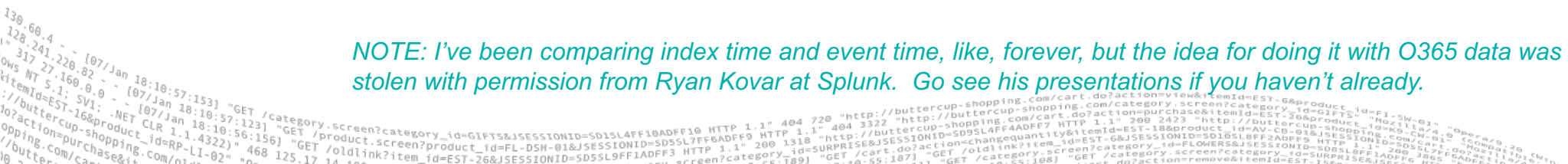
Unfortunately, I Still Have Several Slides Left

- And even though “big, complex systems = big, complex possibilities”…
  - (Foreshadowing)
- …we still need to look at some use case examples, don’t we.

Use Case: Traffic

- You don't own these services, nor do you own the first few hops between them and you
- Simple option: monitor known indexes/sourcetypes
  - This one doesn't even get a query
- More complex option: Start looking into logging delay
  - Determine averages, outliers:
    index=[INDEX] sourcetype=ms:* | eval time=_time | eval itime=_indextime | eval lag=(itime - time)/60 | stats avg(lag), min(lag),median(lag), max(lag) by sourcetype
  - Once you have an average indexing delay you're comfortable with, keep an eye out for delays that go over it:
    index=[INDEX] sourcetype=ms:* | eval time=_time | eval itime=_indextime | eval lag=(itime - time)/60 | where lag > [THRESHOLD] | dedup sourcetype | eval index_time=strftime(itime,"%m/%d/%y %H:%M:%S") | eval event_time =strftime(time,"%m/%d/%y %H:%M:%S")| table sourcetype event_time index_time lag

NOTE: I've been comparing index time and event time, like, forever, but the idea for doing it with O365 data was stolen with permission from Ryan Kovar at Splunk. Go see his presentations if you haven't already.



splunk> .conf18

Use Case: Traffic

O365 traffic delay

sourcetype	avg(lag)	min(lag)	median(lag)	max(lag)
ms:o365:management	18.23894049735868	0.7929999987284343	11.350313070042887	1440.0542339166007
ms:o365:reporting:messagetrace	513.3569352307741	480.60206372	513.75144153	545.21124217

O365 traffic threshold

sourcetype	event_time	index_time	lag
ms:o365:management	09/06/18 20:06:54	09/07/18 13:31:13	1044.3166666666666

Use Case: Login Activity

- Fun quirks: internal vs. external users
  - Internal userids: set up in patterns based on factors in AAD, typically look like what you'd expect
    - user@domain.TLD, user@ms_account_identifier; easy to spot and work with
  - External userids: based on the email address invited to use your org's resource, chopped up with some flavor added in by MS
    - user_domain.TLD#ext#@ms_account_identifier
    - Requires more legwork to normalize
- Normalizing external UserIds
  - sourcetype=ms:* UserId=*#ext#*
    | rex field=UserId "(?<FullExtUser>[^#]+)"
    | rex field=FullExtUser "^(?<username>[\w\W]+)_"
    | rex field=FullExtUser "(_)(?<external_domain>[^_]+)$"
    | eval user=username. "@" .external_domain
  - Identify, get rid of external identifier, grab username, grab external domain, stick them together

Use Case: Login Activity

- Once your UserIds are normalized, you can actually work with them
  - Together or separately, your choice – might depend on factors like userbase, team's goals from the data, etc.
  - Example: external login activity is parsed down to the user and domain levels, why not do something with that?
    - A dashboard with two input fields, USER and DOMAIN, allows you to search for username/domain combos
      - sourcetype=ms:o365* UserId="*#ext#*" Operation=*Login*
        | iplocation ClientIP
        | rex field=UserId "(?<FullExtUser>[^#]+)"
        | rex field=FullExtUser "^(?<username>[\w\W]+)_"
        | rex field=FullExtUser "(_)(?<external_domain>[^_]+)$"
        | eval user=username. "@" .external_domain
        | search user=$username$@$external_domain$
        | table _time Workload user ClientIP Country
        | rename Workload AS App user AS User ClientIP AS "Source IP"

Use Cases: Login Activity

Apr through Jul, 2018 | * | * | Submit | Hide Filters

O365 External User Logins

_time	App	User	Result
2018-06-27 15:29:56	AzureActiveDirectory	[redacted].com	Succeeded
2018-06-18 15:22:29	AzureActiveDirectory	[redacted].com	Succeeded
2018-06-18 07:40:41	AzureActiveDirectory	[redacted].co.uk	Succeeded
2018-06-17 23:05:44	AzureActiveDirectory	[redacted].com	Succeeded
2018-06-17 21:28:44	AzureActiveDirectory	[redacted]	Succeeded

Use Case: Email Activity

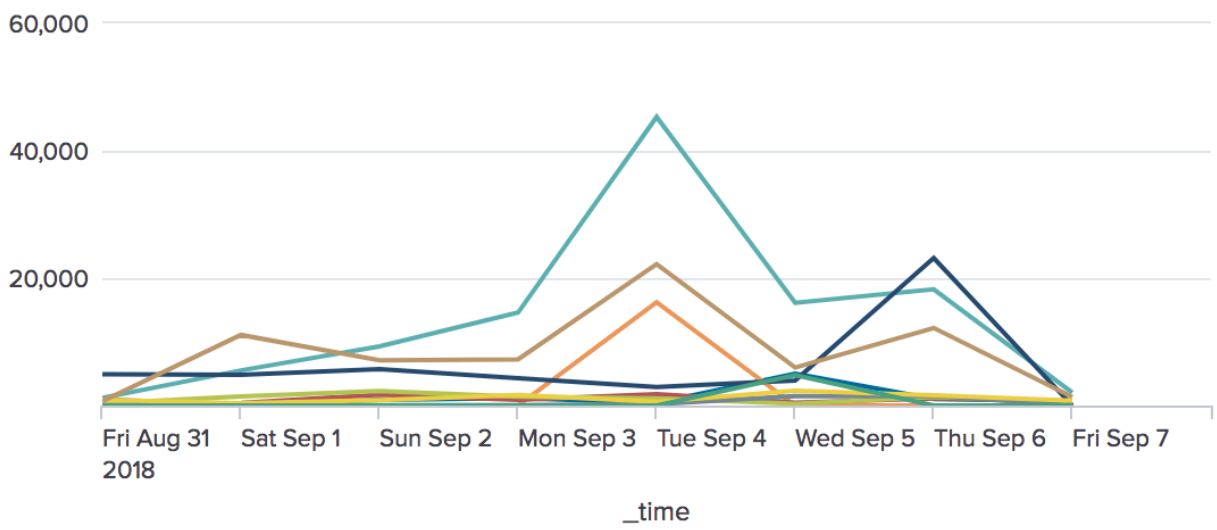
- Where?
  - Client use: sourcetype=ms:* Workload=Exchange
    - “User A moved message B to folder C”
    - The boring stuff, IMO
  - Email tracing: sourcetype=ms:o365:reporting:messagetrace
    - “Email A sent from user B to user C was received at time D containing contents E….”
    - Where the fun stuff is
    - Spam: sourcetype=ms:o365:reporting:messagetrace action=FilteredAsSpam | stats count by SenderDomain | sort –count
      - Also, action=Failed or action=Quarantined for anomalous activity or sanity checks
    - High sender/recipient domains:
      - Incoming by sender: sourcetype=ms:o365:reporting:messagetrace | eval RecipientDomain=lower(RecipientDomain) | search RecipientDomain=[YOUR DOMAIN] SenderDomain!=[YOUR DOMAIN]| eval SenderDomain=lower(SenderDomain) | timechart count by SenderDomain limit=10 useother=false
      - Outgoing by recipient: sourcetype=ms:o365:reporting:messagetrace | eval RecipientDomain=lower(RecipientDomain) | search SenderDomain=[YOUR DOMAIN] RecipientDomain!=[YOUR DOMAIN]| eval SenderDomain=lower(SenderDomain) | timechart count by SenderDomain limit=10 useother=false

splunk> .conf18

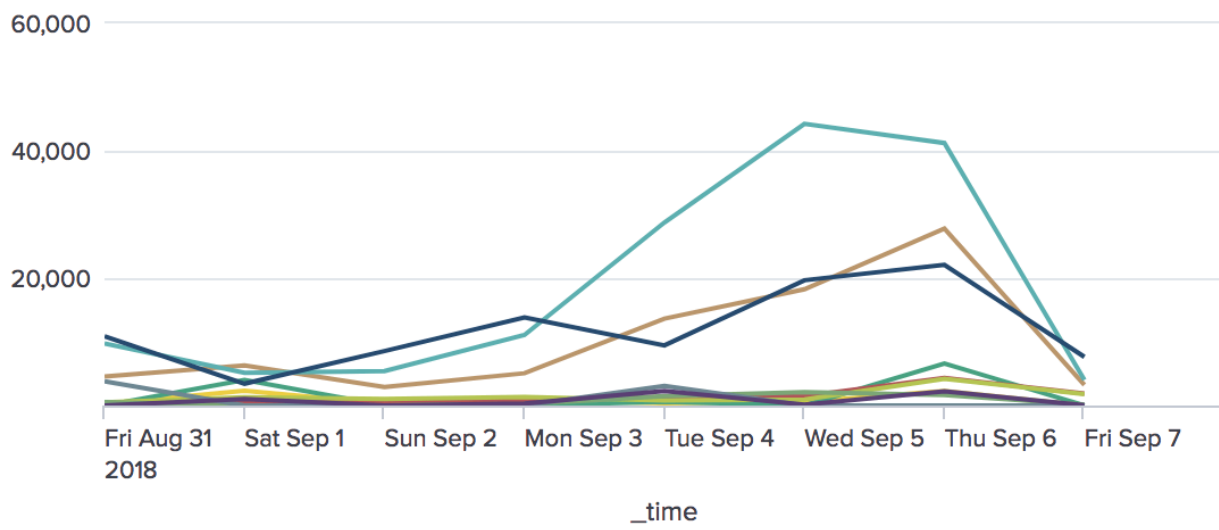
Use Cases: Email Activity

Time: Last 7 days | User: * | Domain: * | Submit | Hide Filters

Email: Top Sender Domains



Email: Top Recipient Domains, outgoing



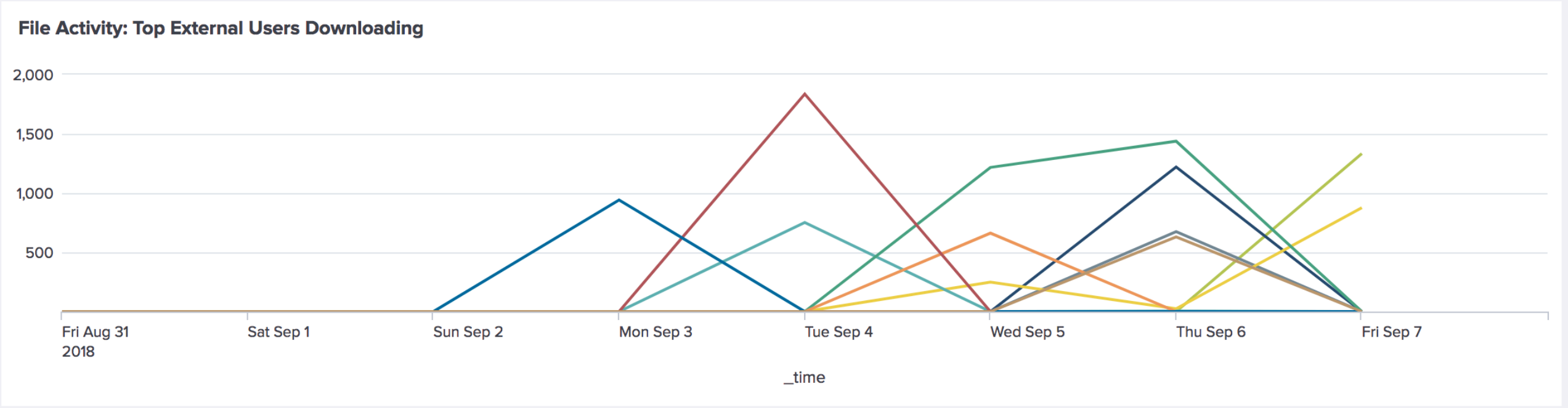
Use Case: File Sharing Activity

- Ton of things you can do here
  - Onedrive + Sharepoint primary sources
  - Lots of activity, some of it irrelevant
    - e.g. sync activity: I want to know when a person syncs a new device, but seeing 1 event for every single file that gets synced is a bit much
- Example use case: Files downloaded by external users
  - Useful for tracking potential exfil attempts
  - Build on previous examples: filter external users, normalize, sort results
    - sourcetype=ms:o365* UserId="*#ext#*" Operation=*FileDownload*
      | iplocation ClientIP
      | rex field=UserId "(?<FullExtUser>[^#]+)”
      | rex field=FullExtUser "^(?<username>[\w\W]+)_"
      | rex field=FullExtUser "(_)(?<domain>[^_]+)$”
      | eval user=username. "@" .domain
      | timechart count by user limit=10 useother=false

splunk> .conf18

Use Cases: File Sharing Activity

Time: Last 7 days | User: * | Domain: * | Submit | Hide Filters



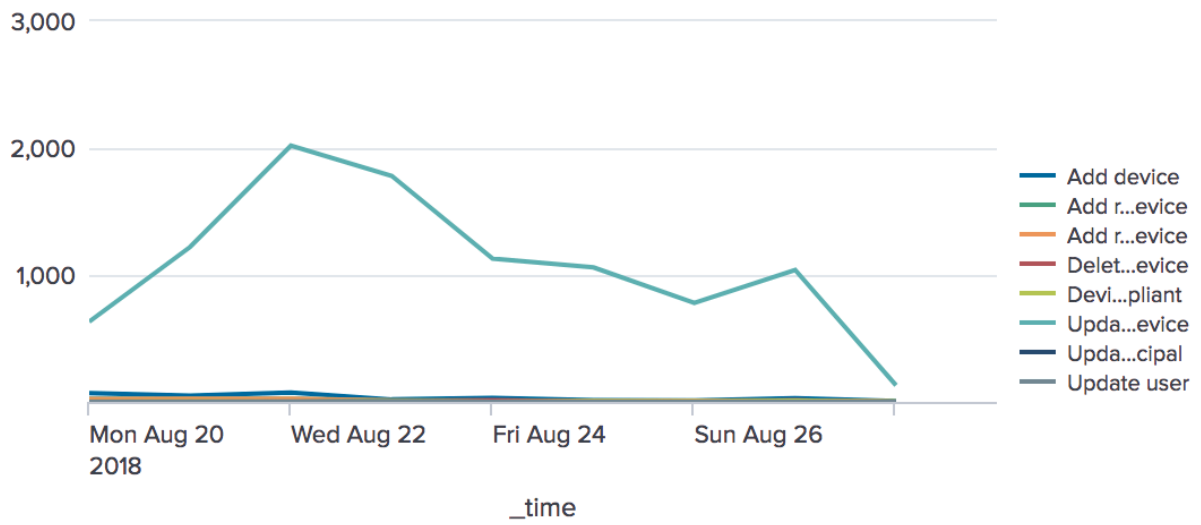
Use Case: Mobile Device Enrollment

- Weird edge case example
  - MS uses Intune to register and track use of mobile devices
  - Request from team member: “Why isn’t Intune data in the O365 logs?”
    - Answer: Because it’s an Azure service, not O365
  - But it doesn’t come in through Azure Monitor…
    - And audit log functionality isn’t even available to send to an Event Hub in Azure…
    - Solution: It actually comes in through the Microsoft Azure Active Directory Reporting Add-on for Splunk
  - Activity by activity type:
    - sourcetype=ms:aad:* actor.name = "Microsoft Intune" OR actor.name = "Device Registration Service”
      | timechart count by activity
  - Activity by user:
    - index=azure sourcetype=ms:aad:* actor.name = "Microsoft Intune" OR actor.name = "Device Registration Service"
      activity!="Update Device" targets{}.userPrincipalName=*
    - | rename targets{}.userPrincipalName AS user
    - | timechart count by user

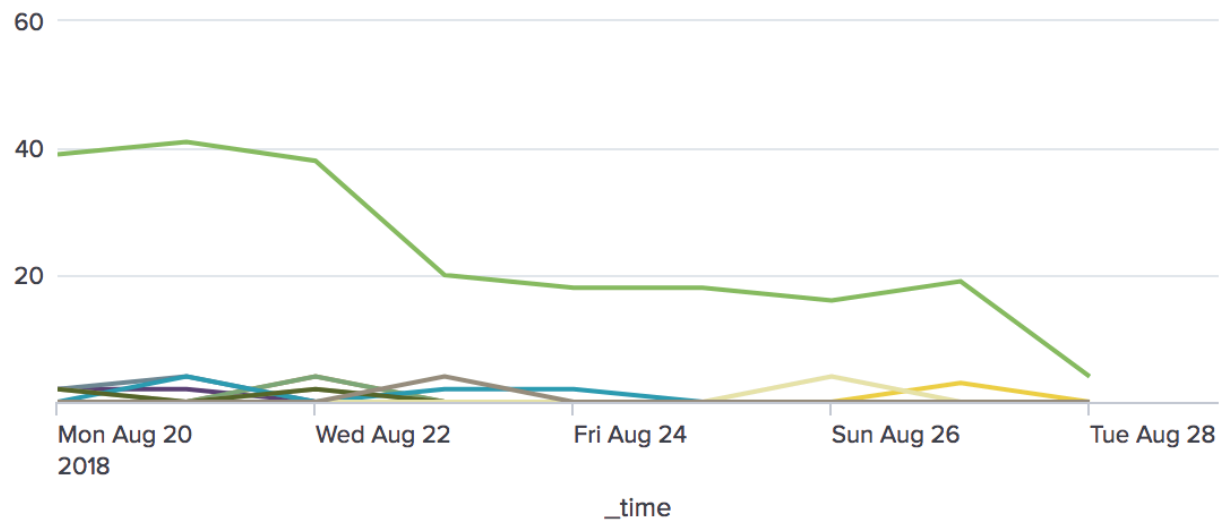
Use Case: Mobile Device Enrollment

Time: Aug 20 through 27, 2018 | User: * | Domain: * | Submit | Hide Filters

Intune Service Activity



Intune Activity By User



splunk> .conf18

Point Seven: Know Your Use Cases

- First level bullets should be sentence case, 24pt
  - Second level bullets, 20pt
    - Third level bullets, 20pt
      - Fourth level bullets, 16pt

Fifth level tab is for highlighted text, 20pt

Sixth level tab is for highlight content, 28pt, Bold

Seventh level is for paragraphs with no bullets, 24pt

Eighth level is for paragraphs with no bullets, 20pt

Ninth level is for paragraphs with no bullets, 16pt

PART IV: Wrap-up

I need a nap

So, What Did We Learn?

1. It’s doable
2. It takes a minute
3. But it’s useful

Further Resources

- Splunking Microsoft Cloud data, parts 1-3
  - https://www.splunk.com/blog/2017/07/27/splunking-microsoft-cloud-data-part-1.html
  - Seriously, do this one first
- Splunking Azure Monitor data, parts 1-2
  - https://www.splunk.com/blog/2018/04/20/splunking-microsoft-azure-monitor-data-part-1-azure-setup.html
  - Like above, but for TA-Azure_Monitor

MS Docs

- Microsoft's TA-Azure_Monitor guide
  - https://github.com/Microsoft/AzureMonitorAddonForSplunk/wiki/Azure-Monitor-Addon-For-Splunk#documentation
  - Another perspective on installation and configuration
- AMQP 1.0 in Azure Service Bus and Event Hubs protocol guide
  - https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-amqp-protocol-guide
  - AMQP (TA-Azure_Monitor's delivery protocol) documentation from MS
  - When you're arguing with your firewall admins about AMQP ports, you'll thank me

.conf Presentations

- 2018:
  - IT1164 – Gain End-to-End Visibility into your Azure Cloud Environment Using Splunk, Jason Conger, Splunk
  - SEC1297 – Down in the Weeds, Up in the Cloud: Splunking Your Azure and Office 365, Ryan Lait, Splunk
  - SEC1355 – Hunting the Known Unknown: Microsoft Cloud, Ryan Kovar and Steve Brant Splunk
    - All of these were yesterday. Whoops. Hit up the recordings after .Conf
- 2017:
  - Monitor And Manage Your Cloud Environment With Azure Monitor And Splunk, John Kemnetz, Microsoft Azure
    - https://conf.splunk.com/files/2017/slides/monitor-and-manage-your-cloud-environment-with-azure-monitor-and-splunk.pdf

Thank You

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