# Building an Asset Inventory Framework with Splunk

Phil Meyerson
Baltimore Splunk Usergroup
4-15-19

# Disclaimer

- Thoughts and opinions expressed are my own and do not represent my employer.
- Understand your data before you act on it.

# Agenda

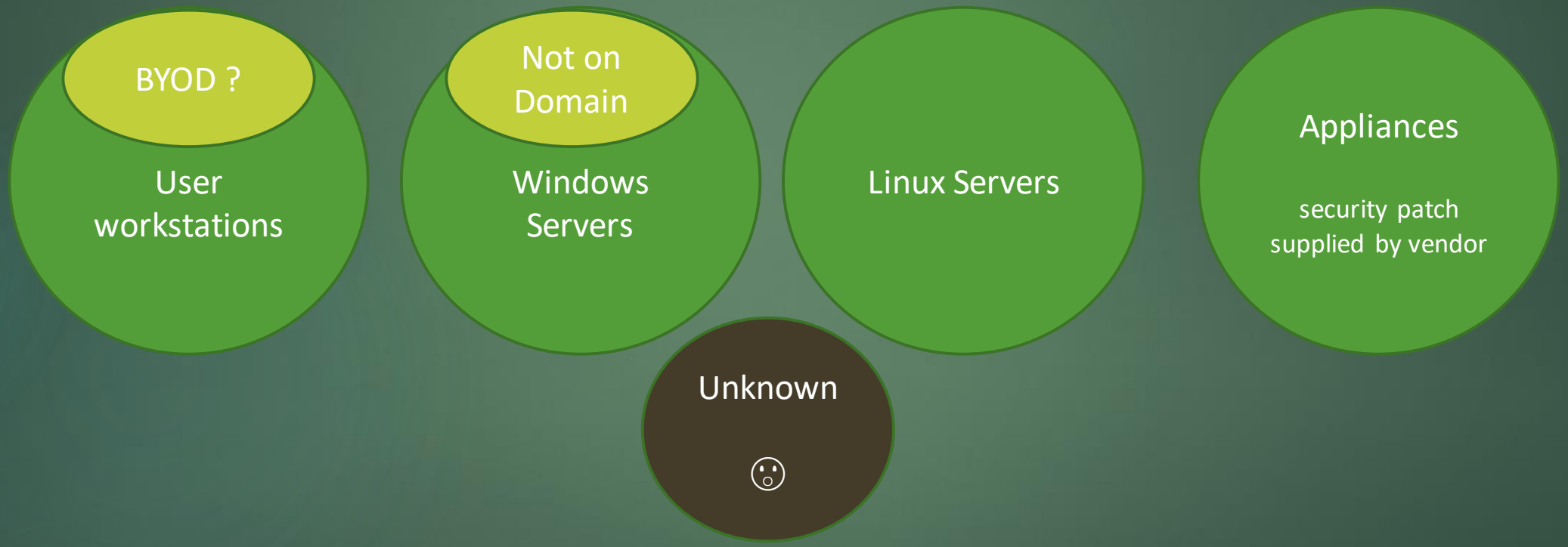
- Introduction
- Solution
  - Framework Overview
  - Build Out Components
- Useful Dashboards & SPL
- Do as I say not as I did

# Inspiration

"… NEVER knowing how many servers there are, virtual machines, endpoints devices, what's covered by what scope-wise. And it's one of these fundamental problems in security that for some reason is really obvious, and many of us have lived with this pain, but nobody's really solved yet."

- Patrick Heim, Operating Partner and CISO, ClearSky, RSAC Innovation Lab judge

# What's on our Network?



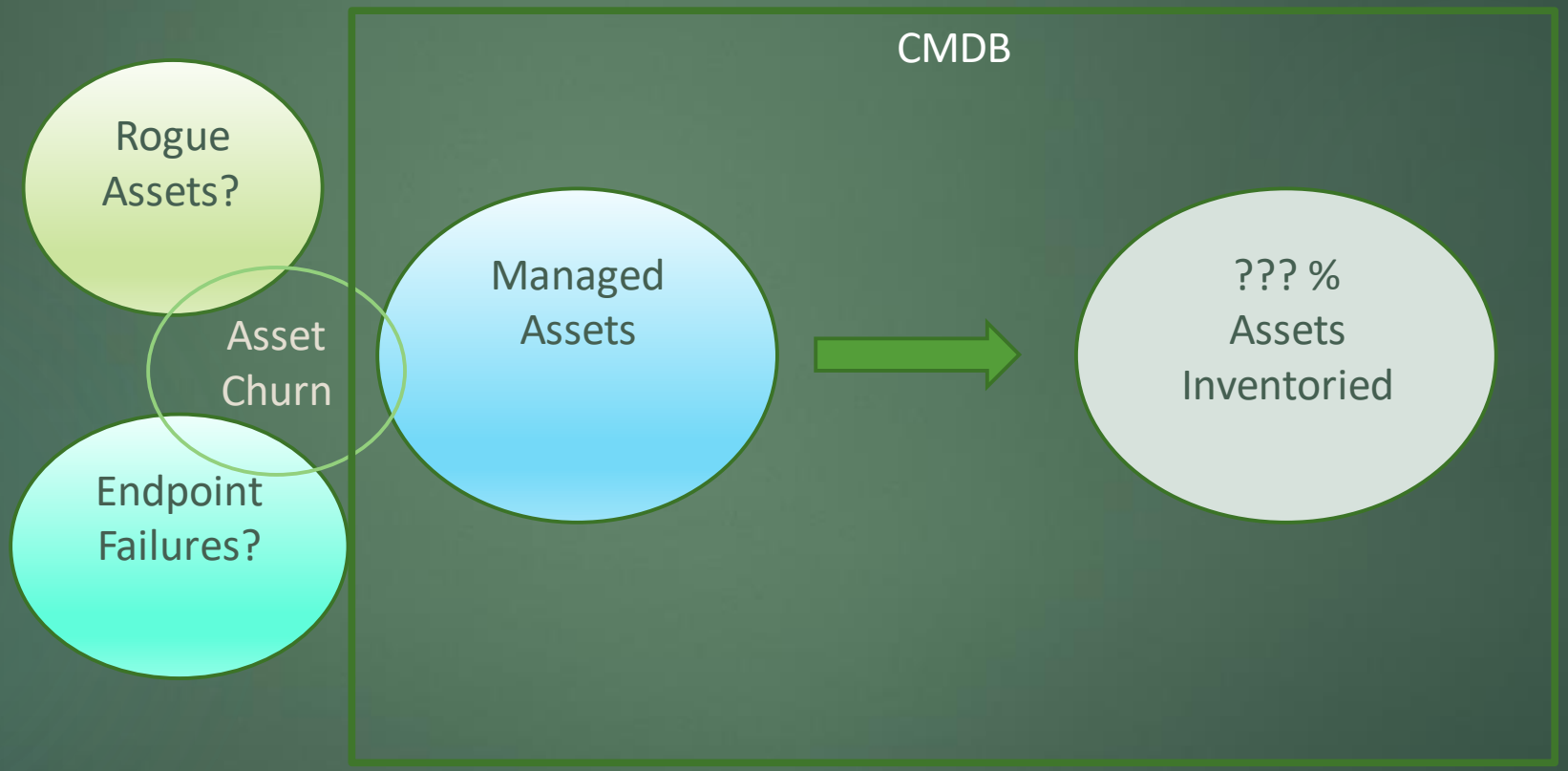
# Why Do We Care?

- Do the basics better to keep improving Operational & Security posture
- Provide data to help teams manage assets better

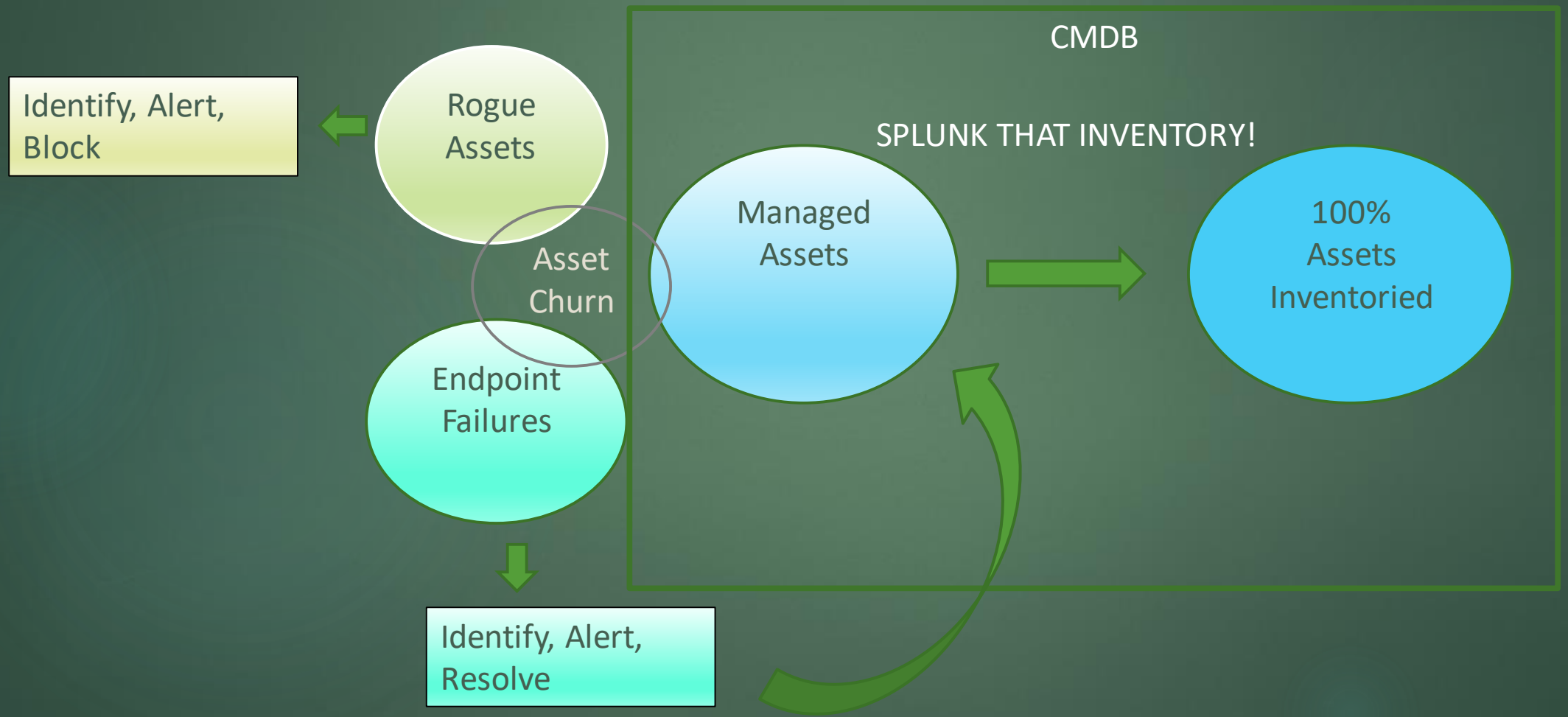
# Objectives

- Identify all assets on corporate network
- Identify assets missing agent(s)
- Identify crud metadata that needs to be purged
  - Provide data to counter cleanup fear

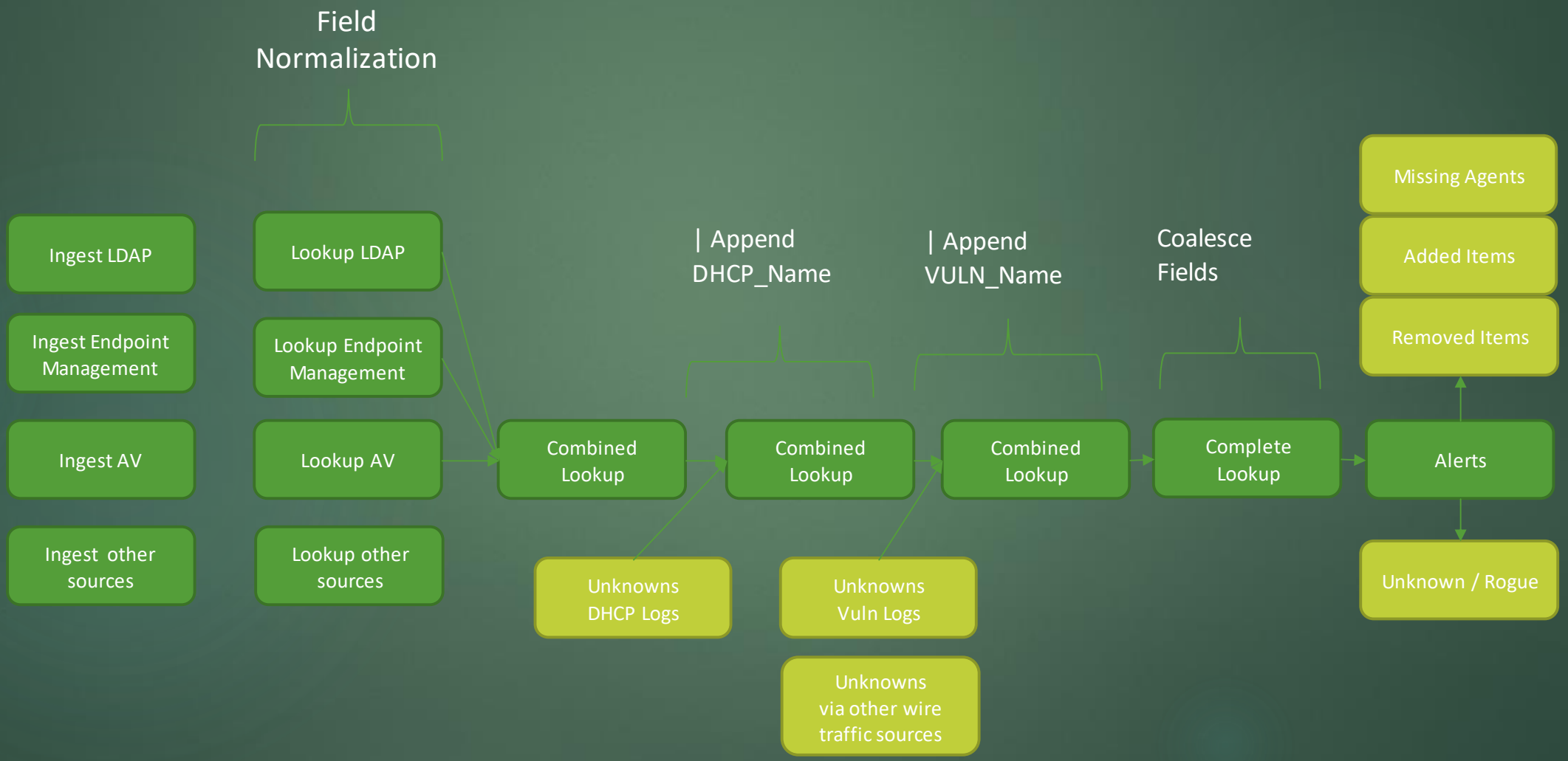
# Challenges


CMDB
Rogue Assets?
Asset Churn
Endpoint Failures?
Managed Assets
??? % Assets Inventoried

# Goal


CMDB
SPLUNK THAT INVENTORY!
Identify, Alert, Block
Rogue Assets
Asset Churn
Managed Assets
100% Assets Inventoried
Endpoint Failures
Identify, Alert, Resolve

# Inventory Framework



# Inventory Framework

- Mix in whatever sources you have at your disposal
- Preferably include both asset tracking sources as well as wire traffic  or scans to provide a complete picture.
- Very easy to start simple and grow as you mature.  No need to complete all steps at once.

# End Result (partial)

| AD_Name | Domain | AD_LastActive | AD_CreationDate | OperatingSystem | CMDB_Classification | CMDB_InstallStatus | MGMT_ClientVersion |
|---|---|---|---|---|---|---|---|
| XXX | CONTOSO | 2018-06-11 21:15:42 | 2013-10-21 14:19:12 | Windows Server 2008 R2 Standard | Production | In Stock | Null |
| XXX | CONTOSO | 2019-04-16 17:42:57 | 2019-04-16 17:42:56 | Windows Server 2012 R2 Standard | Production | Installed | 5.00.8577.1003 |
| XXX | CONTOSO | 2019-04-10 20:38:49 | 2019-03-11 19:17:10 | Windows Server 2016 Standard | Production | Installed | 5.00.8577.1003 |

# The Pieces

- SA-LdapSearch (Supporting Add-On for Microsoft Windows)
  - Use | collect to summary index* for historical tracking capability
- TA-rest (REST Modular Input)
  - Antivirus Agent API
  - CMDB API
  - May need a bit of python skills
- DBX (DB Connect)
  - Endpoint Management System database
- Saved Searches
  - Nightly snapshot from each data source reports
  - Comprehensive list of domains

# Field Normalization at search time

- Consistent naming syntax:
  - CamelCase, or SOURCE_CamelCase  -- helpful for |table *Name*
- Consistent field names for common fields
  - Hostname = name / cn / host
  - LastActive = last logon time, last checkin, last update, etc
  - InstallDate = join to LDAP time, agent install time
- If done at index time… don’t change your mind later.
- Consistent datetime format, ISO-8601 works well for sorting

# rest_ta Input Service Now

URL: https://domain.cmdbservice.com/api/table/cmdb_ci_server?
?sysparm_display_value=true
METHOD: GET
AUTH: xxx
Response Type: json
Response Handler: CustomResponseHandler
Polling Interval: 5 1 * * * (1:05 AM)
[x] Index Error Responses
sourcetype: cmdb:server
index=inventory

Additional inputs:
/cmdb_ci_win_server; sourcetype = cmdb:win
/cmdb_ci_computer; sourcetype = cmdb:workstation

# CMBB Lookup

```
index=inventory sourcetype=cmdb:win
| fields - _raw, *.link
| dedup name, sourcetype
| append
    [ search index=inventory sourcetype=cmdb:server NOT
        [ search index=inventory sourcetype=cmdb:win
        | fields name
        | dedup name]
    | dedup name, sourcetype ]

| eval dns_tld = mvindex(split(lower(dns_domain), "."),0)
| eval Os_tld = mvindex(split(lower(os_domain), "."),0)
| eval coalesce_domain=coalesce(Os_tld, dns_tld)

| table …

| fillnull value="Unknown"
| eval asset.display_name=if(isnull(asset.display_name), "Unknown", asset.display_name)
| eval department.display_name=if(isnull(department.display_name), "Unknown", department.display_name)
| eval u_auto_reboot=if(isnull(u_auto_reboot), "Unknown", u_auto_reboot)
| outputlookup CMDB_inventory_servers.csv
```

# Partial inventory_snow.csv

| SNOW_Name | SNOW_Domain | SNOW_VmName | SNOW_ActiveDate | SNOW_ChassisType | SNOW_Classification | SNOW_Department | SNOW_Hard |
|---|---|---|---|---|---|---|---|
| XXX | CONTOSO | XXX | 03-06-2019 02:24:02 PM | Other | Production | Operations | VMware, Inc Virtual Platfo |
| XXX | Unknown | XXX | 09-26-2016 09:38:21 PM | Other | Production | Finance | VMware, Inc Virtual Platfo |
| XXX | Unknown | XXX | 09-25-2017 09:39:07 PM | Other | Production | Finance | VMware, Inc Virtual Platfo |
| XXX | Unknown | XXX | 03-10-2017 09:37:49 PM | Other | Production | Finance | VMware, Inc Virtual Platfo |

# SA-LDAPSearch

```
| ldapsearch search="(objectClass=computer)"
attrs="distinguishedName,cn,lastLogonTimestamp,whenCreated,operatingSystem,
canonicalName"
| eval domain="XX"
| eval ReportDate = strftime(now(), "%Y-%m-%d %H:%M:%S")
| rex field=whenCreated
  "(?<year>\d{4})(?<month>\d{2})(?<day>\d{2})(?<hour>\d{2})(?<min>\d{2})(?<sec>\d{2})"
| eval creationDate = year + "-" + month + "-" + day + " " + hour +":" + min + ":" + sec
| eval lastLogonEpoch=strptime(lastLogonTimestamp, "%Y-%m-%dT%H:%M:%S.%NZ")
| eval pathElements=split(canonicalName, "/")
| eval container=mvjoin(mvindex(pathElements,0,(mvcount(pathElements)-2)), "/")
| collect index=inventory sourcetype=inventory:AD
```

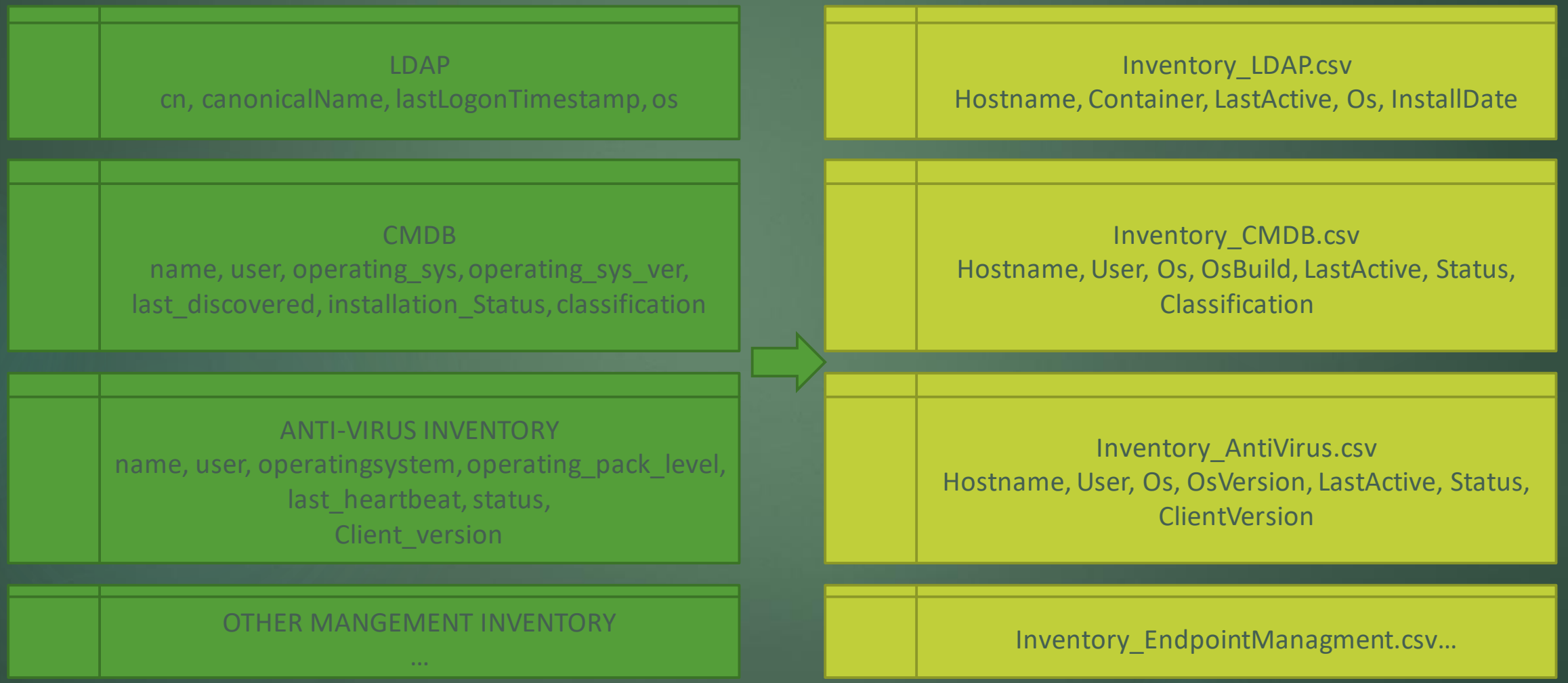
# SA-LDAPSearch

```
...
|rename cn as Hostname, lastLogonTimestamp as LastActive, creationDate as
            InstallDate, domain as Domain, operatingSystem as OperatingSystem

| append
  [... | ldapsearch for each child domain ]

|table Hostname, Domain, OperatingSystem, Container, LastActive, InstallDate,
        ReportDate
| outputlookup inventory_ldap_computers.csv
```

# Partial inventory_ldap.csv

| LDAP_Container | LDAP_Domain | LDAP_Hostname | LDAP_InstallDate | LDAP_LastActive |
|---|---|---|---|---|
| Contoso.com/Servers | CONTOSO | XXX | 2019-04-16 17:42:56 | 2019-04-16 17:42:57 |
| Contoso.com/Servers | CONTOSO | XXX | 2019-04-09 12:35:10 | 2019-04-09 12:35:10 |
| Contoso.com/Servers | CONTOSO | XXX | 2019-03-11 19:17:10 | 2019-04-10 20:38:49 |
| Contoso.com/Servers | CONTOSO | XXX | 2019-03-11 17:53:03 | 2019-04-10 20:45:10 |
| Contoso.com/Servers | CONTOSO | XXX | 2019-03-11 16:16:34 | 2019-04-10 21:02:21 |

# Build Your Lookups



# Build Your Lookups

Inventory_LDAP.csv
Hostname, Container, LastActive, Os, InstallDate

Inventory_CMDB.csv
Hostname, User, Os, OsBuild, LastActive, Status, Classification

Inventory_AntiVirus.csv
Hostname, User, Os, OsVersion, LastActive, Status, ClientVersion

Inventory_EndpointManagment.csv…

Inventory_combined.csv

LDAP_Hostname, CMDB_Hostname, AV_Hostname, CMDB_User, AV_User, LDAP_Os, AV_Os, CMDB_Os, CMDB_OsBuild, AV_OsVersion, LDAP_Container, LDAP_LastActive, CMDB_LastActive, AV_LastActive, CMDB_Status, AV_Status, AV_ClientVersion

# Refine your Lookup

Inventory_combined.csv

LDAP_Hostname, CMDB_Hostname, AV_Hostname, CMDB_User, AV_User, LDAP_Os, AV_Os, CMDB_Os, CMDB_OsBuild, AV_OsVersion, LDAP_Container, LDAP_LastActive, CMDB_LastActive, AV_LastActive, CMDB_Status, AV_Status, AV_ClientVersion

Inventory_combined.csv

Hostname, User, Os, OsBuild, OsVersion, Container, LDAP_LastActive, CMDB_LastActive, AV_LastActive, CMDB_Status, AV_Status, AV_ClientVersion,…

# Refine: Examine Data Quality

- Check for quality of field values
  - | stats values(*name*), values(*fqdn*), values(*host*)
- Check for field coverage
  - | stats count by (*name*), (*fqdn*), (*host*)

# Refine: Examine Data Quality

- May require different coalescence rules for different portions of data
  - EX: LDAP has most detailed “Os” values… except when os is linux… then use CMDB or another source.
  - | eval OperatingSystem =
    case(like(LDAP_Os, "Microsoft Windows%"), LDAP_Os,
    MGMT_Os=="Null", LDAP_Os,
    MGMT_Os=="none", LDAP_Os,
    1==1, MGMT_Os)

# Refine: Examine Data Quality

```
|inputlookup combined_inventory.csv
| eval consistent=if(
    lower(CMDB_Fqdn)==lower(CMDB_Name) AND
    lower(CMDB_Fqdn)==lower(CMDB_Hostname), "true",
"false")
|search consistent=“false”
| table *
```

# Refine: Examine Data Quality

- |inputlookup
  |stats count(*fqdn*), count(Hostname) count(name) count(*dns*)

| | | | |
|---|---|---|---|
| 1826 | 1817 | 1999 | 1752 |

But are some fields more detailed than others?

```
|inputlookup
| stats values(*fqdn*), value(Hostname), values(name), values(u_Vmname)
```

# Refine: Examine Data Quality

Inventory with non matching values across fqdn, name, and Hostname:

```
|inputlookup
| eval consistent=if(lower(fqdn)==lower(name) AND
                     lower(fqdn)==lower(host_name), “True", “False")
|fillnull value=“UNKNOWN”
|search NOT(fqdn=UNKNOWN AND host_name=UNKNOWN)
| table sourcetype, fqdn, name, host_name, dns_domain, consistent
| search consistent=False
```

---

cmdb:win server1verylongname server1verylong server1verlongname

# Combined Inventory View

| LDAP_Hostname | LDAP_Domain | LDAP_LastLogonTime | LDAP_CreationDate | OperatingSystem | CMDB_Classification | CMDB_InstallStatus | MGMT_ClientVersion | AV_Status | AV_Version | AV_LastSeen | LDAP_Contain |
|---|---|---|---|---|---|---|---|---|---|---|---|
| XXX | CONTOSO | 2019-04-14 23:55:50 | 2012-08-06 16:01:23 | Windows Server 2008 R2 Standard | Production | Installed | 6.03 | true | 6.1.7.10741 | 2019-04-18 12:46:00 | Contoso.com/ |
| XXX | CONTOSO | 2019-04-17 21:14:51 | 2013-11-06 19:05:34 | Windows Server 2008 R2 Enterprise | Production | Installed | 6.03 | true | 6.1.7.10741 | 2019-04-18 12:46:01 | Contoso.com/ |

# Combined Inventory View

```
| inputlookup inventory_ldap_computers.csv
| rename * as LDAP_*
| lookup inventory_EndpointManagement.csv Hostname as LDAP_Hostname OUTPUT
ClientVersion as MGMT_ClientVersion, LastActive as MGMT_LastActive …
|lookup inventory_av.csv name as LDAP_Name OUTPUT clientversion as
            AV_ClientVersion, lastHeartbeat as AV_LastActive ….
| fillnull value=“Null”
| eval OperatingSystem =
    case(like(LDAP_Os, "Microsoft Windows%"), LDAP_Os,
        MGMT_Os=="Null", LDAP_Os,
        MGMT_Os=="none", LDAP_Os,
        1==1,MGMT_Os)
```

# Combined Inventory View 2

```
| eval unmanagedOU =`unmanagedOU(LDAP_Container)`
| eval DisabledOU=`disabledOU(LDAP_Container)`
| eval CutoffDate = relative_time(now(), $cutoff_age$)
| eval PastCutoff = if(LDAP_LastActive > CutoffDate, “True”, “False”)
| table LDAP_Name, *Name*, OperatingSystem, *Domain*, *Active*, *Install*,
        *Version* LDAP*, MGMT*, CMDB*, AV*, DisabledOU, UnmanagedOU,
         PastCutoff
| outputlookup inventory_combined.csv
```

# Add Unknowns from Scans

```
| inputlookup inventory_combined.csv
| append
        [ search index=inventory sourcetype=vuln_scan
              NOT [|inputlookup inventory_combined.csv
                        | fields Name | rename  Name as Hostname ]
          | dedup Hostname, src_ip
          | rename Hostname as VULN_Hostname, src_ip as VULN_Ip ]
| table *
| outputlookup inventory_combined_vulnscan.csv
```

# Add Unknowns from DHCP

```
|inputlookup inventory_combined_tenable.csv
| append
        [ search index=dhcp sourcetype=dhcp dhcp_type=DHCPACK
            NOT [|inputlookup inventory_combined_vulnscan.csv
                | fields dest_hostname, dest_mac ]
        | dedup dest_Hostname, dest_mac
        | rename dest_Hostname as DHCP_Name, dest_mac as DHCP_Mac ]
 | table *
 | outputlookup complete_inventory.csv
```

# Helpful Dashboards

- | REST /services/data/lookup-table-files
  | search title="*inventory*"
  | rename eai:appName as AppName, title as LookupName
  | table AppName LookupName

| AppName | LookupName |
| --- | --- |
| search | inventory_ldap_computers.csv |
| search | inventory_endpoint_management.csv |
| search | inventory_domains.csv |
| search | inventory_cmdb_servers.csv |

# Display Data Refresh Time

```
| inputlookup inventory_MGMT_computers.csv
| head 1
| eval source="MGMT"
| append
      [| inputlookup inventory_ldap_computers.csv
       | head 1
       | eval source="LDAP"
       | table ReportDate, Source]
| append    [ ... ]
| table ReportDate, Source
```

| ReportDate | Source |
|---|---|
| 2019-04-15 | MGMT |
| 2019-04-15 | LDAP |
| 2019-04-15 | CMDB |
| 2019-04-15 | AV |

# Count by Domain, Source

| LDAP Count | Endpoint Management Count | CMDB Count | Domain |
|---|---|---|---|
| 2 | 2 | 2 | XXX |
| 629 | 296 | 424 | XXX |
| 0 | 74 | 34 | XXX |
| 986 | 288 | 17 | XXX |
| 0 | 0 | 1 | XXX |
| 2521 | 2142 | 77 | XXX |
| 1464 | 693 | 34 | XXX |
| 0 | 0 | 2 | XXX |
| 0 | 0 | 0 | XXX |
| 0 | 0 | 1 | XXX |
| 0 | 0 | 92 | XXX |
| 0 | 1 | 0 | XXX |
| 0 | 1 | 0 | XXX |
| 0 | 3 | 1 | XXX |
| 5602 | 3500 | 685 | XXX |

# Count by Domain, Source

```
| inputlookup inventory_domains.csv
| join domain
      [| inputlookup inventory_ldap_computers.csv
       | eval domain=lower(domain)
       | stats count by domain
       | append
             [| inputlookup inventory_domains.csv
              | search NOT
                    [| inputlookup inventory_ldap_computers.csv
                     | eval domain=lower(domain)
                     | stats values(domain) as domain
                     | mvexpand domain]
              | eval count=0]
       | rename count as “LDAP Count"]
| join domain     [… for each source ]
|addcoltotals
```

List all domains

Stats count by domain

Append 0 as count for domains not in above inputlookup

# Count by Domain, Source

- Otherwise the stacked | join commands will result in loss of rows
- Technique referred to as “Sentinel Values” – 0s to pad the stats table, see Starcher, Waddle 2015 conf Lookup Talk “Beyond the Lookup Glass”

# Min Logon Age by OU

| Container | Most Recent Login | Oldest Login | count |
|---|---|---|---|
| red.com/disabled workstations (do not delete)/03_12_2019 | 2019-04-13 21:07:41 | 2010-02-15 05:46:53 | 536 |
| fabrikim.com/disabled workstations (do not delete)/03_12_2019 | 2019-04-01 12:48:41 | 2012-01-24 20:19:50 | 373 |
| fabrikim.com/disabled workstations (do not delete)/remove objects 2_26_19 | 2019-04-10 15:22:15 | 2009-02-14 15:25:12 | 255 |
| blue.com/computers | 2025-07-01 08:18:20 | 2010-10-22 13:24:29 | 226 |
| contoso.com/disabled workstations (do not delete)/objects to be removed 03_06_2019 | 2019-04-01 17:57:13 | 2016-01-11 18:35:39 | 123 |
| red.com/disabled workstations (do not delete)/03_04_2019 | 2015-04-01 12:09:14 | 2009-04-15 19:11:21 | 107 |
| contoso.com/disabled workstations (do not delete)/objects to be removed 3_26_2019 | 2019-04-03 18:37:38 | 2018-11-29 13:30:20 | 52 |

Aid in cleanup efforts!
Caveat: Replication delay in LDAP time field!
Filtered to “unmanaged” or “marked for deletion” containers
Paranoid? Filter asset names by LDAP_Container (as above), then subsearch to find most recent login timestamp from authentication logs. Or check if all systems disabled.

# Min Logon Age by OU

```
| inputlookup inventory_ldap_computers.csv
| rename …    | fillnull value="Null“
| eval LastActive =if(LastActive ="Null", InstallDate, LastActive)
| eval unmanagedOU=`unmanagedOU(Container)`
| eval disabledOU=`disabledOU(Container)`
| search unmanagedOU=True OR disabledOU=True
| where InstallDate < relative_time(now(), "-30d@d")

| stats max(LastActive) as “Most Recent Login”,
        min(LastActive) as “Oldest Login”,
        count by Container
```

# `removed_items(5)`

Find inventory difference between two time deltas.  We specify the star time, the end time, and a time that separates the two.

EX: `removed_items(combined_inventory, LDAP_Name,
“04/05/2019:00:00:00”, “04/06/2019:00:00:00”, “04/07/2019:00:00:00”)`

index=inventory sourcetype=$st$ earliest=$first_time$ latest=$middle$ NOT

[ search index=inventory sourcetype=$st$ earliest=$middle$ latest=$last_time$

| fields $host_value$]

$st$ = sourcetype inspecting
$host_value$ = field containing Hostname
$first_time$ =  the initial timeframe
$middle$ = the end of the initial timeframe.  Also the start of the ending timeframe

$last_time$ = the end of the ending timeframe

# Custom rest_ta handler

```
class CustomResponseHandler:
# response from API is in format {"result": [{}, {}, {}]} at least for CMDB tables; add to responsehandler.py
# credit http://www.georgestarcher.com/splunk-null-thinking/

  def __init__(self,**args):
      pass

  def __call__(self, response_object, raw_response_output, response_type, req_args, endpoint):
      #if response_type == "json":
      output = json.loads(raw_response_output)

      for entry in output['result']:
          clean_entry = {k: v for k, v in entry.items() if v} ← AWESOME
          print_xml_stream(json.dumps(clean_entry))
```

# Auth from Unknown Host!

```
index=msad* EventCode=4776 Source_Workstation=*
| eval Source_Workstation=trim(Source_Workstation,"\\")
| search NOT
        [| inputlookup combined_inventory.csv
         | fields LDAP_Hostname
         | rename LDAP_Hostname as Source_Workstation]
| stats values(User), count by Source_Workstation
```

# Example Report Schedule

- Ingest: 00:00 – 01:00
- Lookups: 01:00 – 02:00
- Combined View: 02:00 – 03:00
- Complete View: 04:00 – 05:00
- Alerts 06:00 – 07:00

# Keys to Success

- Pull all relevant data from sources and index at least daily.
- Give yourself capability to drill in on a single asset details to quickly vet data and dashboards
- Dig into your data – plenty of traps – search for expertise
  - Non-replicated AD fields
  - Confusingly-named fields in SCCM
- More than 10,000 assets? Sub-searches will fail!
  - Break up inventory grouping by Domain, OU, Department, etc.
- Design search filters to account for data quality
  - Old objects not being purged from LDAP
  - Management process moving objects to OU and retain for xxx days

# Mistakes I made

- Don’t use the ldap path attribute with = (EX: cn=computers,dc=contoso,dc=com)…. You will not be able to pull out of summary index easily!
- Use consistent naming conventions across lookups and dashboards
  - Standardize field names in nightly lookups
  - Prepend source to field name in combined view
- Field normalization is easiest at search time!
- Use macros and consistent naming of fields

# References


- http://www.georgestarcher.com/splunk-null-thinking/
- https://nmap.org/book/man-hOst-discovery.html
- https://conf.splunk.com/session/2015/conf2015-LookupTalk.pdf
- DB Connect - https://splunkbase.splunk.com/app/2686/
- rest_ta* - https://splunkbase.splunk.com/app/1546/
- Supporting Add-on for Active Directory https://splunkbase.splunk.com/app/1151/
- Simple NMAP* - https://splunkbase.splunk.com/app/3056/
- Splunk Add-on for Tenable - https://splunkbase.splunk.com/app/1710/
- Docs.splunk.com, answers.splunk.com, splunk-Usergroups.slack.com!
- https://medium.com/axonius/the-toyota-camry-of-cybersecurity-axonius-wins-rsac-2019-innovation-sandbox-to-solve-the-asset-18b1b69f0125


* Denotes Community Supported App