Forward-Looking Statements

This presentation may contain forward-looking statements regarding future events, plans or the expected financial performance of our company, including our expectations regarding our products, technology, strategy, customers, markets, acquisitions and investments. These statements reflect management's current expectations, estimates and assumptions based on the information currently available to us. These forward-looking statements are not guarantees of future performance and involve significant risks, uncertainties and other factors that may cause our actual results, performance or achievements to be materially different from results, performance or achievements expressed or implied by the forward-looking statements contained in this presentation.

For additional information about factors that could cause actual results to differ materially from those described in the forward-looking statements made in this presentation, please refer to our periodic reports and other filings with the SEC, including the risk factors identified in our most recent quarterly reports on Form 10-Q and annual reports on Form 10-K, copies of which may be obtained by visiting the Splunk Investor Relations website at www.investors.splunk.com or the SEC's website at www.sec.gov. The forward-looking statements made in this presentation are made as of the time and date of this presentation. If reviewed after the initial presentation, even if made available by us, on our website or otherwise, it may not contain current or accurate information. We disclaim any obligation to update or revise any forward-looking statement based on new information, future events or otherwise, except as required by applicable law.

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. We undertake no obligation either to develop the features or functionalities described or to include any such feature or functionality in a future release.

splunk> .conf21

Effective & Affordable Cyber-Security Vulnerability Management With Splunk Enterprise

SEC1075A

Daniel Ferreira

Vulnerability Management Lead
Global Cyber-Security Team | Shell IT International B.V.

splunk> .conf21

What if you can create your own ‘natural language’ search engine?

```
`asset` conf.splunk.com
| table asset_owner asset_business_unit asset_exposure asset_exposed_ports
```

```
`asset` (AWS OR Azure) virtual machine internet facing
| table asset_name asset_ad_domain asset_ad_memberof asset_installed_agent
```

```
`asset` tcp (80 OR 443 OR 8443) internet facing (BigIP OR Big-IP OR TMUI)
| table a*name a*type a*owner a*country* a*ports a*last_seen a*service
```

```
`asset` "solarwinds agent"
| table a*name a*type a*owner a*platform a*last_logged_on
```

```
`software` apache struts 2.* internet facing
| table asset_name a*exposure software_vendor software_name s*_version
```

```
`identity` (domain OR enterprise) admins
| table identity_forest i*_domain i*_name i*_owner i*_memberof i*_created i*logon*
```

```
`mobile` noncompliant (iPad OR iPhone) 14.*
| table mobile_imei mobile_platform mobile_model mobile_os mobile_owner mobile_last_seen
```

```
`vulnerability` CVE-2021-34527 PrintNightmare (remediated OR open)
| table asset_name vulnerability_unique_id v*remediation_status v*first_seen v*last_seen
```

Daniel Ferreira

Vulnerability Management Lead
Global Cyber-Security Team
Shell IT International B.V.

splunk> .conf21

The team

1st

Biggest European company by revenue (5th world-wide)

4

IT Hubs across the world in 3 continents

100+

Cyber Security professionals, global in-house team

TB+

Multiple Terabytes per day on Splunk ingestion

Agenda

1) Effective Vulnerability Management
   How to do a decent job without vulnerability scanners

2) Asset & Software inventory
   Enumeration of assets & software, data sources

3) Data ingestion
   TERM() & PREFIX() ready data

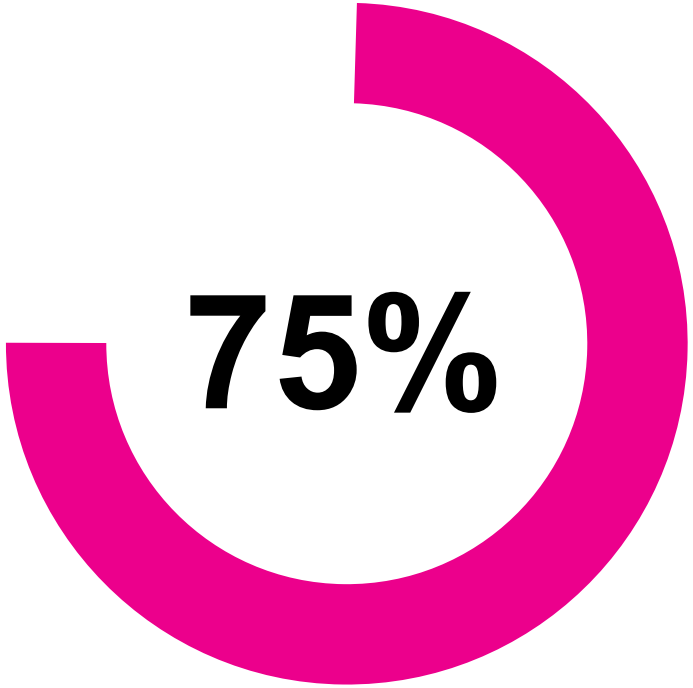
4) Data processing
   Cooking the data to make it even more TERM() available

5) Data querying
   Use the asset inventory to track vulnerabilities

6) Real world examples of usage

splunk> .conf21

Asset inventory and Software inventory value weight on Vulnerability Management



It is all about knowing what you have

Our experience says up to 75% of cyber-security vulnerability management is just about a solid **Asset inventory** & **Software inventory**. This means you only need vulnerability scanners for the remaining 25% of vulnerabilities and to automatically follow up on all the universe over time.

Asset inventory sources

Machine-generated only sources, they all produce a valid **Hostname**

Always avoid ‘human generated’ asset inventory data (aka spreadsheets).

Scanners

Traditional appliance-based scanners

nmap

nmap ‘discovery-only’ scripts for isolated networks or in case you don’t have scanners

Cloud APIs

Enumeration of VMs, storage, load balancers, etc.

Agents

EDR agent, AV agent, scanning agent, all other agents you can read from

Active Directory

Enumeration of AD domain trust, later proceed with enumeration of Computers per domain

CMDB is not asset inventory

Instead, consider CMDB as ‘a source’ to feed scanners

Avoid following sources to be considered “line-item” generators:

- Excel files.
- CSV files.
- SQL dumps from systems that do not ‘keep state’.
- “Configuration Items” or “CIs”.

Instead, integrate these into Splunk and scan them via scanners (or nmap).

This technique will guarantee your asset inventory “line-items” are always alive assets.

However

Merge problem

A lot of **|inputlookup**, **|append** or **|join** commands are required. This solution is slow and does not scale on asset inventories with millions of rows and daily refreshes.

The PREFIX() directive

Making all your data **|tstats** ready

(with just Splunk Enterprise or Splunk Cloud Platform)

The PREFIX() directive game-changer

PREFIX() = Ability to **retrieve** terms from .tsidx!

- Before Splunk 8, we were only capable of **querying** terms with the **TERM()** directive in **|tstats**, but not having them back to the pipeline (retrieve them).
  - In 'OR' conditions, you never knew which term you hit.
  - E.g. TERM(process=powershell.exe) OR TERM(process=cmd.exe)

Data Models are not needed.

Note there is no 'FROM' clause!

```
| tstats
    latest(PREFIX(scheduled_time=)) AS latest_schedule_time
    WHERE
    index=_internal
    source=/opt/splunk/var/log/splunk/scheduler.log
    earliest=@d
    TERM(daniel.ferreira@secret.com)
    TERM(important-scheduled-report-name)
    BY
    PREFIX(status=)
    PREFIX(success=)
| rename *= AS *
```

✓ **656 events** (7/23/21 12:00:00.000 AM to 1/19/38 4:14:07.000 AM) No Event Sampling

Events (656) Patterns **Statistics (1)** Visualization

100 Per Page Format Preview

status	success
delegated_remote_completion	1

Data source format matters

JSON, CSV, plain text log?

```
1  {
2      "dnshostname": "WSAMSW89102.addomain.local",
3      "cn": "WSAMSW89102",
4      "lastlogontimestamp": "2021-07-15T09:50:07.6943274+02:00",
5      "operatingsystemversion": "10.0 (18901)",
6      "operatingsystem": "Windows 10 Enterprise"
7  }
```

Double quote major breaker? No **PREFIX()** for you! Sorry! 😢

What if we can 'enforce' some key fields to be TERM() and PREFIX() ready in JSON format?

Introducing the ‘dotnotation’ property

Making all JSON data sources **PREFIX()**-ready

```
{
    "dnshostname": "WSAMSW89102.addomain.local",
    "cn": "WSAMSW89102",
    "lastlogontimestamp": "2021-07-15T09:50:07.6943274+02:00",
    "operatingsystemversion": "10.0 (18901)",
    "operatingsystem": "Windows 10 Enterprise",
    "dotnotation": [
        "cn=WSAMSW89102",
        "dnshostname=WSAMSW891025.addomain.local"
    ]
}
```

Guaranteeing presence of key fields in **PREFIX()**-ready format.

- Asset name, hostname, FQDN or any other DNS information is always preferable.
- Asset ID is not mandatory as it will vary per source system.
- IP address only if no DNS datapoint is seen.

- Not meant for ‘all properties’, **just ‘key fields’**.
- Trade offs:
  - Some bytes more per event on license.
  - Some bytes more per event on storage.
  - Processing compute power (cheap alternative: server-less).
  - ⚠️ Be careful with prefixes clashes.

Get **ConvertTo-DotNotation** PowerShell cmdlet from Github.

ConvertTo-SplunkHEC: PowerShell cmdlet

Because we 💙 languages with pipelines!

```
PS C:\
>  # Asset inventory from scanner, with dotnotation and converted to HEC payload:
>> ,(Import-Csv E:\data\scanner_assets.csv `
>> | Select-Object *, @{n='dotnotation';e={$_ | Select-Object asset_id, host_name, ip_address | ConvertTo-DotNotation}}) `
>> | ConvertTo-SplunkHEC -IndexValue scanner -SourcetypeValue asset:inventory -SourceValue 'asset_id=(?<source>[\w\-\.]+)' -SourceValueIsRegex -HostValue 'ip_address=(?<host>[\w\-\.]+)'
 -HostValueIsRegex -OutputArrayChunks 10000
```

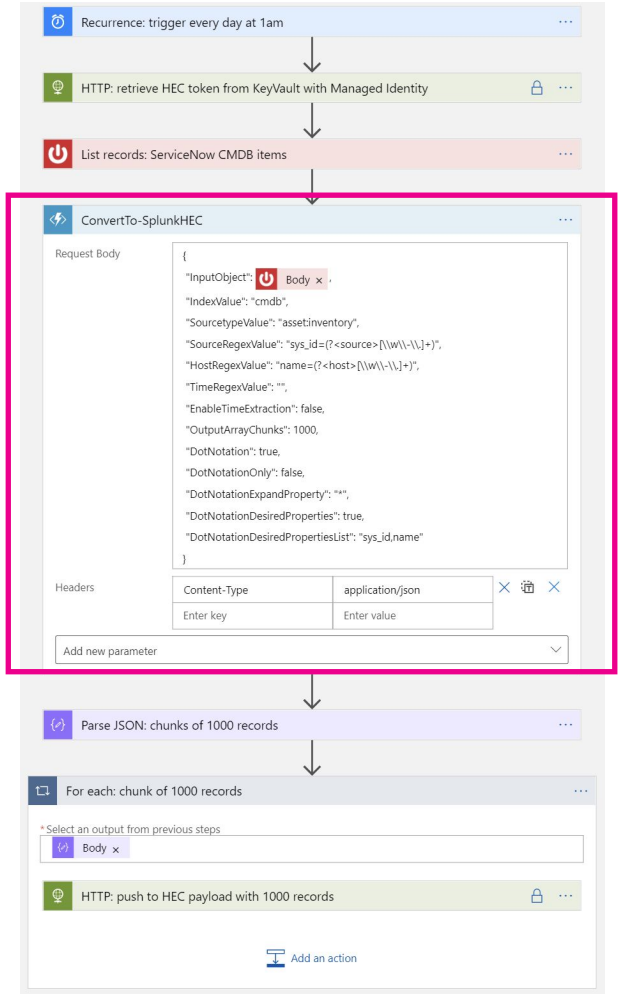

```
[
    [
        {
            "event": {
                "asset_id": "613135",
                "ip_address": "10.45.244.21",
                "host_name": "WSAMSW891025.addomain.local",
                "os_name": "Windows",
                "os_version": "10.0 (18901)",
                "os_description": "",
                "os_system": "Microsoft Windows",
                "last_assessed_for_vulnerabilities": "07/12/2021 18:20:20",
                "dotnotation": [
                    "asset_id=613135",
                    "host_name=WSAMSW891025.addomain.local",
                    "ip_address=10.45.244.21"
                ]
            },
            "index": "scanner",
            "sourcetype": "asset:inventory",
            "source": "613135",
            "host": "10.45.244.21",
            "time": 1627059093
        }
    ]
]
```


Splunk HEC ready payload with **'dotnotation'** object.

Get [ConvertTo-SplunkHEC](#) PowerShell cmdlet from Github.

Cloud APIs integrations

Leveraging the compute power of PaaS serverless integration services



Cloud-native integration tools help to ‘glue’ 3rd party systems together.

- **Azure Logic Apps with Managed Identities**
  - Ideal for batch-based Cloud integration.
  - Not only for Azure-based data. Anything that has an API can be plugged in.
- **Azure Functions / AWS Lambda / Google Functions**
  - Ideal for streaming-based Cloud integrations.
  - Most likely paired with other streaming tools: Event Hubs, Kinesis, etc.
- **Look for affordability:** money is 👑, you can integrate high-volume & high-speed data with few 💵.

Get ConvertTo-SplunkHEC (serverless) PowerShell cmdlet from Github.

Serverless version contains **ConvertTo-DotNotation** cmdlet.

splunk> .conf21

Alive assets: your asset inventory contains assets seen ‘alive’ over the last **2 months.**

Because you only import ‘alive assets as of yesterday’, per source, per day; you’re naturally embedding ‘asset decay’ for assets not showing up over the period.

```
| tstats
    fillnull_value=null
    latest(_time) AS asset_last_seen_epoch

    ``` CMDB: ```
    latest(PREFIX(sys_id=)) AS cmdb__sys_id

    ``` Scanner: ```
    latest(PREFIX(asset_id=)) AS scanner__asset_id

    WHERE
    (index=cmdb OR index=scanner OR index=ad)
    sourcetype=asset:inventory
    earliest=-2mon

    BY
    PREFIX(name=)          ``` name:        DNS name coming from CMDB ```
    PREFIX(cn=)            ``` cn:          DNS name coming from Active Directory ```
    PREFIX(host_name=)     ``` host_name:   DNS name coming from Scanner ```
                           ``` Why not 'host' field? Because it can have different meaning or unexpected value: e.g. UF hostname. ```
| rename *= AS *

``` Convert "null" to real null() to enable coalesce to work: ```
| foreach name,cn,host_name [ | eval <<FIELD>>=if(match(<<FIELD>>,"^null$"), null(), <<FIELD>>) ]

``` Coalesce fields across data sources: ```
| eval asset_id=coalesce(cmdb__sys_id, scanner__asset_id)
| eval asset_name=coalesce(name, cn, host_name)

``` Enforce NetBIOS-equivalent DNS name: ```
| rex field=asset_name "(?<asset_name>[^.]*)"

``` Merge: ```
| stats max(asset_last_seen_epoch) AS asset_last_seen_epoch, values(asset_id) AS asset_id BY asset_name
```

✓ 3 events (7/22/21 8:37:11.000 PM to 1/19/38 4:14:07.000 AM) No Event Sampling ▾ Job ▾ Fast Mode ▾

Events Patterns Statistics (1) Visualization

100 Per Page ▾ Format Preview ▾

asset_name	asset_last_seen_epoch	asset_id
wsamsw89102	1627063915	2de56707148c2e303fbebb645267a284613135
```

Asset Enrichment Feeds

1) CMDB
   - Business, Sub-business, Service provider, Owner, Deployment, Class, Status, etc.
2) Cloud tenants: Azure, AWS, Google
   - Platform, Owner, Owner by action, Last seen, ID, Local users, Installed software, etc.
3) Agents (EDR, AV, sysmon, etc):
   - OS, Installed software, Installed KBs, Missing KBs, Last reboot, Logged on, etc.
4) Active directory:
   - Last seen, Memberof, Forest, Domain, OUs, custom properties, etc.
5) Custom pulls (local scripts, UF scripts, etc):
   - Installed KBs, Last reboot, Installed software, Local users, etc.
6) Custom scans:
   - HTTP Headers, HTML Title, Redirect URLs, etc.
7) Any other reliable asset information...

splunk> .conf21

However

Merge problem again 😟

How do we merge all of these data repositories? What to use as '**unique key**' field?

Answer: we do not need a single 'unique key', instead we can have '**many possible unique keys**' 😉

Introducing multi-value: asset_key

All possible ‘unique identifiers’ for an IT asset

😍 **Multi-value** fields allow you to do many-to-many joins with **|lookup** command.

asset_key
3951416
912f79254b8f234bac78c584b3c9e9b2
10.31.21.77
52.11.193.95
ec2amaz-priceapp1
ec2-52-11-193-95.ap-southeast-1.compute.amazonaws.com
ec2-52-11-193-95
www.pricingapp.domain.com
i-0f88f1ee8f16eea41
ip-10-31-21-77.ap-southeast-1.compute.internal

Scanner ID.

EDR Agent ID.

Private IP address.

Public IP address.

AWS EC2 Instance Name.

AWS EC2 Public IP FQDN and short DNS.

Domain Name ‘A record’ resolution.

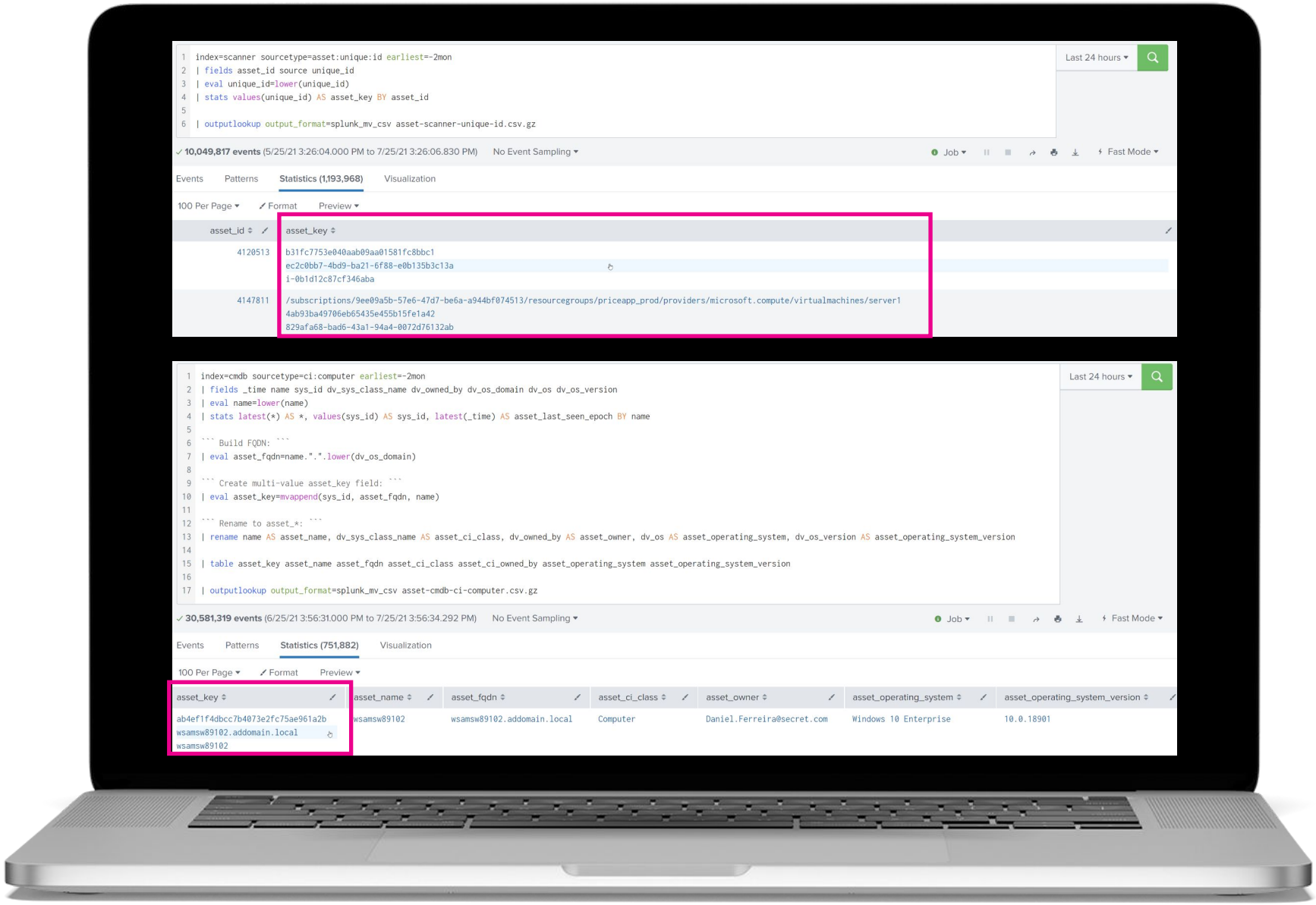
AWS EC2 Instance ID.

AWS EC2 Private IP FQDN.

Integration lookups

Each enrichment source will have its own lookup

Prepare all asset enrichment data in **independent lookup files**, all with its very unique '**asset_key**' multi-value field.



```
25
26  ``` Coalesce fields across data sources: ```
27  | eval asset_id=coalesce(cmdb__sys_id, scanner__asset_id)
28  | eval asset_name=coalesce(name, cn, host_name)
29
30  ``` Enforce NetBIOS-equivalent DNS name: ```
31  | rex field=asset_name "(?<asset_name>[^.]*)"
32
33  ``` Merge: ```
34  | stats max(asset_last_seen_epoch) AS asset_last_seen_epoch, values(asset_id) AS asset_id BY asset_name
35
36  ``` asset_key definition: ```
37  | eval asset_key=mvappend(asset_id, asset_name)
38
39  ``` Bring other possible unique-identifiers for the asset when they have agent installed. Coming from scanner, assets with agents installed: ```
40  | lookup asset-scanner-unique-id.csv.gz asset_id AS asset_key OUTPUT asset_key AS temp_scanner_unique_id__asset_key
41  | eval asset_key=mvdedup(mvappend(asset_key, temp_scanner_unique_id__asset_key))
42
43  ``` Bring CMDB enrichment fields: ```
44  | lookup asset-cmdb-ci-computer.csv.gz asset_key OUTPUT asset_key AS temp_cmdb__asset_key, asset_ci_class asset_owner asset_operating_system asset_operating_system_version
45  | eval asset_key=mvdedup(mvappend(asset_key, temp_cmdb__asset_key))
46
47  | table asset_key asset_name a*
```

asset_key: initial definition using all unique identifiers coming from the ‘line-item generation command’ and DNS name.

Many-to-many match: search in each lookup by all possible values seen in ‘asset_key’ field. Then, bring all enrichment fields.

Add more values to ‘asset_key’: each enrichment source may have detected other unique identifiers for the asset, so bring them all into ‘asset_key’ too.

✓ **3 events** (7/22/21 5:11:14.000 PM to 1/19/38 4:14:07.000 AM) No Event Sampling ▾ Job ▾ Fast Mode ▾

Events Patterns **Statistics (1)** Visualization

100 Per Page ▾ Format Preview ▾

asset_key	asset_name	asset_ci_class	asset_id	asset_last_seen_epoch	asset_operating_system	asset_operating_system_version	asset_owner
2de56707148c2e303fbebb645267a284613135 wsamsw89102 wsamsw89102.addomain.local	wsamsw89102	Computer	2de56707148c2e303fbebb645267a284613135	1627063915	Windows 10 Enterprise	10.0.18901	Daniel.Ferreira@secret.com

Few months later...

(actually not as happy as these faces)

😑

```
    ```
    ### ACTIVE DIRECTORY:
    ```
| lookup asset-active-directory.csv.gz asset_key OUTPUT asset_domain_cn, asset_ad_domain, asset_ad_distinguished_name, asset_ad_last_logon_timestamp, asset_operating_system AS
    temp_ad__asset_os, asset_operating_system_version AS temp_ad__asset_os_version, asset_ad_memberof
| eval asset_operating_system=coalesce(temp_ad__asset_os, asset_operating_system)
, asset_operating_system_version=coalesce(temp_ad__asset_os_version, asset_operating_system_version)
, asset_operating_system_source=if(isnull(asset_domain_cn), asset_operating_system_source, "AD:Computers")   ``` 👀: define first field 'asset_operating_system_source'.  ```
, asset_operating_system_certainty=if(isnull(asset_domain_cn), asset_operating_system_certainty, "1")         ``` 👀: define first field 'asset_operating_system_certainty'. ```
, asset_is_unlocked=if(isnull(asset_domain_cn), asset_is_unlocked, if(match(lower(asset_ad_distinguished_name),"ou=admin rights") AND match(lower(asset_ad_distinguished_name),"ou
    =workstations"), "true", asset_is_unlocked))                                                              ``` 👀 : detect admin-rights workstations. Define the field first.```
, asset_is_domain_joined=if(isnull(asset_domain_cn),"false","true")
```

AD Groups this asset belongs to: remember this for later...

```
    ```
    ### EDR:
    ```
| lookup asset-edr.csv.gz asset_key OUTPUT asset_key AS temp_edr__asset_key, asset_os_name AS temp_edr__asset_os_name, asset_os_version AS temp_edr__asset_os_version
| eval asset_operating_system=coalesce(temp_edr__asset_os_name, asset_operating_system)
, asset_operating_system_version=coalesce(temp_edr__asset_os_version, asset_operating_system_version)
, asset_operating_system_source=if(isnull(temp_edr__asset_key), asset_operating_system_source, "EDR")
, asset_operating_system_certainty=if(isnull(temp_edr__asset_key), asset_operating_system_certainty, "1")
, asset_installed_agent=if(isnull(temp_edr__asset_key), asset_installed_agent, mvappend(asset_installed_agent,"EDR-vendor-X"))
, asset_key=if(isnull(temp_edr__asset_key), asset_key, mvdedup(mvappend(asset_key,temp_edr__asset_key)))

    ```
    ### EDR: Last logged on:
    ```
| lookup asset-edr-userlogon.csv.gz asset_key OUTPUT asset_last_logged_on_username AS temp_edr__asset_last_logged_on_username
| eval asset_last_logged_on=mvappend(asset_last_logged_on, temp_edr__asset_last_logged_on_username)

    ```
    ### EDR: Host applied vulnerability kb:
    ```
| lookup asset-edr-installed-kb.csv.gz asset_key OUTPUT asset_installed_vulnerability_kb AS temp_edr__installed_kb
| eval asset_installed_vulnerability_kb=mvappend(asset_installed_vulnerability_kb, temp_edr__installed_kb)

    ```
    ### EDR: Host last reboot::
    ```
| lookup asset-edr-last-reboot.csv.gz asset_key OUTPUT asset_last_reboot_epoch AS temp_edr__asset_last_reboot_epoch
| eval asset_last_reboot_epoch=if(isnull(temp_edr__asset_last_reboot_epoch), asset_last_reboot_epoch, temp_edr__asset_last_reboot_epoch)
```

Installed KBs per machine: thereby the opposite can also be calculated (assets missing certain KB), in conjunction with ‘asset_operating_system*’ fields.

```
88      ```
89      ### Azure Virtual Machines:
90      ```
91  | lookup asset-cloud-azure-vms.csv.gz asset_key OUTPUT asset_key AS temp_az__asset_key, asset_platform AS temp_az__asset_platform, asset_platform_description AS temp_az__asset_platform_description, asset_owner_by_action AS
      temp_az__asset_owner_by_action, asset_owner AS temp_az__asset_owner, asset_installed_agent AS temp_az__asset_installed_agent
92  | eval asset_platform=if(isnull(temp_az__asset_key), asset_platform, temp_az__asset_platform)
93  , asset_platform_description=if(isnull(temp_az__asset_key), asset_platform_description, temp_az__asset_platform_description)
94  , asset_owner=mvdedup(mvappend(asset_owner,temp_az__asset_owner))
95  , asset_owner_by_action=if(isnull(temp_az__asset_owner_by_action), asset_owner_by_action, mvdedup(mvappend(asset_owner_by_action,temp_az__asset_owner_by_action)))    ``` 😟 ```
96  , asset_key=if(isnull(temp_az__asset_key), asset_key, mvdedup(mvappend(asset_key,temp_az__asset_key)))
97  , asset_installed_agent=mvdedup(mvappend(asset_installed_agent, temp_az__asset_installed_agent))
98
99      ```
100     ### AWS Virtual Machines:
101     ```
102 | lookup asset-cloud-aws-vms.csv.gz asset_key OUTPUT asset_key AS temp_aws__asset_key, asset_platform AS temp_aws__asset_platform, asset_platform_description AS temp_aws__asset_platform_description, asset_owner_by_action AS
      temp_aws__asset_owner_by_action, asset_owner AS temp_aws__asset_owner
103 | eval asset_platform=if(isnull(temp_aws__asset_key), asset_platform, temp_aws__asset_platform)
104 , asset_platform_description=if(isnull(temp_aws__asset_key), asset_platform_description, temp_aws__asset_platform_description)
105 , asset_owner=mvdedup(mvappend(asset_owner,temp_aws__asset_owner))
106 , asset_owner_by_action=if(isnull(temp_aws__asset_owner_by_action), asset_owner_by_action, mvdedup(mvappend(asset_owner_by_action,temp_aws__asset_owner_by_action)))    ``` 😟 ```
107 , asset_key=if(isnull(temp_aws__asset_key), asset_key, mvdedup(mvappend(asset_key,temp_aws__asset_key)))
108
```

asset_key	asset_name	asset_platform	asset_platform_description
3951416 912f79254b8f234bac78c584b3c9e9b2 10.31.21.77 52.11.193.95 ec2amaz-priceapp1 ec2-52-11-193-95.ap-southeast-1.compute.amazonaws.com ec2-52-11-193-95 www.pricingapp.domain.com i-0f88f1ee8f16eea41 ip-10-31-21-77.ap-southeast-1.compute.internal	ec2amaz-priceapp1	AWS	AWS Account: SD9100039 (Retail Price App - Prod) AWS Instance name: ec2amaz-priceapp1 AWS EC2 virtual machine instance id: i-0f88f1ee8f16eea41 AWS Public IP: 52.11.193.95

Installed software

Important fields - example

Deduplication: ~~imperfect~~ deduplication of repeated values of software names coming from multiple sources.

Installed software:

- Pulled from custom scripts (e.g. UF PowerShell), EDR, SCCM, Scanner, etc.
  - For asset inventory purposes, keep the ‘software name’ only **without the version number**.

```
110
111     ```
112     ### Installed software products:
113     ### Note: the purpose of asset_installed_software is to show installed products, not software versions; so the asset becomes SEARCHABLE by product names.
114     ```
115     ``` ### Installed software products - EDR: ```
116 | lookup software-edr.csv.gz asset_key OUTPUT software_name AS temp_edr__software
117 | eval asset_installed_software=if(isnull(temp_edr__software), asset_installed_software, mvdedup(mvappend(asset_installed_software,temp_edr__software)))
118     ``` ### Installed software products - Custom pulling script: ```
119 | lookup software-uf-pull.csv.gz asset_key OUTPUT software_name AS temp_uf__software
120 | eval asset_installed_software=if(isnull(temp_uf__software), asset_installed_software, mvdedup(mvappend(asset_installed_software,temp_uf__software)))
121     ``` ### Installed software products - Scanner (agent and remote authenticated scans): ```
122 | lookup software-scanner.csv.gz asset_key OUTPUT software_name AS temp_scanner__software
123 | eval asset_installed_software=if(isnull(temp_scanner__software), asset_installed_software, mvdedup(mvappend(asset_installed_software,temp_scanner__software)))
124
```

Owner By Action

Important fields - example

Keeping difference between owners: officially registered owners should keep their own fields, so ‘discovered’ ones can be identified.

Owner by action:

- Anybody that ‘touches’ a resource within a Cloud Subscription/Account becomes automatically an owner.
  - Need to onboard to Splunk the Audit Logs for all Azure Subscriptions / AWS Accounts.
    (you should onboard this regardless)

asset_key	asset_platform	asset_platform_description	asset_owner	asset_owner_key	asset_comment
11ab9f3e57a7470db9d729200b538a514156479 conq-eh-test 10.201.21.40 conq-eh-test.addomain.local a1277570-cd85-486f-a7ef-c077c00ac4b1	Azure	Azure tenant: aztenant.onmicrosoft.com Azure subscription id: 8180ab5b-d0c3-41a7-8e91-220ed8d721bb Azure subscription name: Conq-App-TEST Azure resource group: ConQ-EH-TEST-521341 Azure virtual machine: ConQ-EH-TEST	unknown	daniel.ferreira@secret.com	Owner-by-action found: owner(s) 'daniel.ferreira@secret.com' found performing administrative actions on Azure Subscription 'Conq-App-TEST' and Resource Group 'ConQ-EH-TEST-521341'.

splunk> .conf21

What are we going to do with this?

We are going to make all this data 'searchable' by writing to a Summary Index

summary index

Documentation / Splexicon / Summaryindex * A B C D E F G H I J K L M N O P R S T U V W

noun

A special index that stores the results of a scheduled report, when you enable summary indexing for the report. Summary indexing lets you run fast searches over large data sets by spreading out the cost of a computationally expensive report over time. To achieve this, the search that populates the summary index runs on a frequent, recurring basis and extracts the specific data that you require. You can then run fast and efficient searches against this small subset of data in the summary index.

Daily scheduled report...

Daily...

Enrichment fields brought up by |lookup commands...

`asset` macro with natural language text...

Freeze asset inventory data daily

Usage of **|collect** command

```
2148
2149      ```
2150      ### Set _time as of "today at the top of the day" and increase seconds each 100 rows to avoid "sub-second" error:
2151      ```
2152  | streamstats count as row
2153  | eval _time=relative_time(now(),"@d")+(row/100)
2154
2155      ```
2156      ### Convert all Multi-value fields into non-mv by using the " | " delimiter.
2157      ```
2158  | foreach asset* [ eval <<FIELD>>=if(mvcount(<<FIELD>>)>1,mvjoin(<<FIELD>>, " | "),<<FIELD>>) ]
2159
2160      ```
2161      ### Table desired fields:
2162      ```
2163  | table asset_key asset*
2164
2165      ```
2166      ### COLLECT:
2167      ```
2168  | collect index=summary source=asset
```

😱 … a few months later …

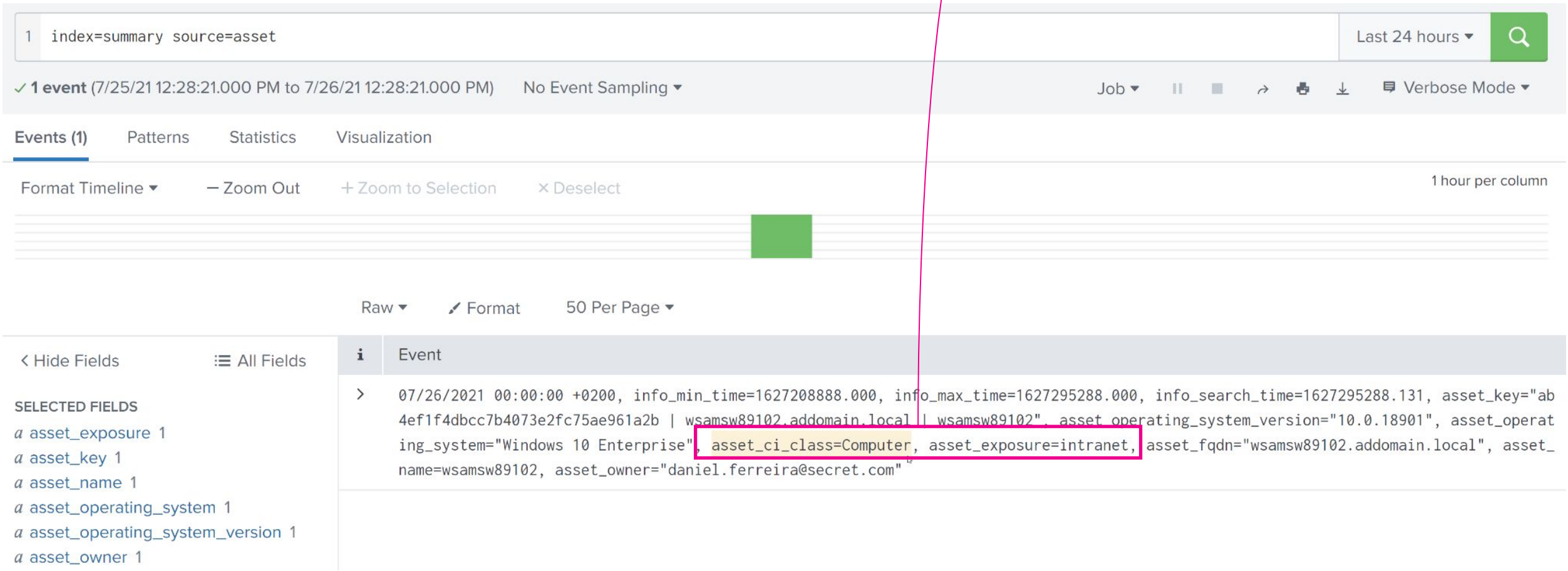
`asset` macro definition:
index=summary source=asset earliest=-01d@d latest=@d

Why the 'earliest' and 'latest'? Because the idea is that Asset Inventory Summary Index gets updated once a day, and all automations required for it will take a while to execute (e.g. AD integration), potentially hours. This timing problem will make ASIA-PAC users unable to search at their morning. So, Asset Inventory in **`asset`** macro is "updated as of yesterday" by default (unless you resolve it manually and move it to earliest=@d). You should also create a KV lookup containing updated data as of today, which then can feed Splunk ES —if you have it— and allow users to **|lookup** it in their other searches.

Summary index results

Stash format, automatic field extraction

💛💙❤️ **TERM() & PREFIX() ready fields:**
Summary Index data is in ‘stash format’, which will make your asset inventory data **|tstats ready**.



"TERM() is used in less than 1% of all customer searches executed on Splunk Cloud."

Richard Morgan
Principal Architect
Splunk

Summary index queries

|tstats all your data with TERM() and PREFIX()

```
| tstats count
    WHERE
    index=summary
    source=asset
    earliest=-1d@d
    latest=@d
    BY
    PREFIX(asset_type=)
| rename *= AS *
```

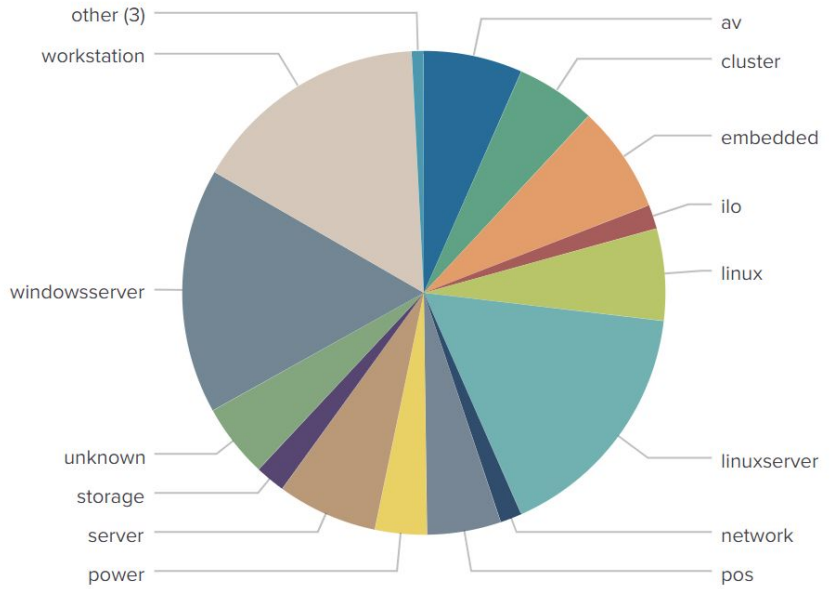
✓ **1,312,195 events** (7/25/21 12:00:00.000 AM to 7/26/21 12:00:00.000 AM)

No Event Sampling

Events | Patterns | **Statistics (17)** | Visualization

100 Per Page | Format | Preview

asset_type
workstation
network
bos
linuxserver
pos
windowsserver




```
| tstats count
    WHERE
    index=summary
    source=asset
    earliest=-30d@d
    latest=@d
    TERM(asset_type=windowsserver)
    BY
    _time span=1d
```

✓ **3,112,626 events** (7/16/21 12:00:00.000 AM to 7/26/21 12:00:00.000 AM)

This search has completed and has returned **30** results by scanning **3,112,626** events in **1.839** seconds

splunk> .conf21

Few real world examples

(actually happy as these faces)

😀

Asset inventory search examples

Use-case: breaching report on an internet facing application

```
1 `asset` pricingapp.domain.com
```

```
1 `asset` 52.11.193.95
```

```
1 `asset` retail app
```

```
1 `asset` aws virtual machine public ip prod*
```

```
1 `asset` i-0f88f1ee8f16eea41 ec2
```

```
1 `asset` 921f79254b8f234bac78c584b3c9e9b2
```

```
1 `asset` aws retail machine
```

```
1 `asset` 10.31.21.77
```

```
1 `asset` sd91*
```

asset_key	asset_name	asset_platform	asset_platform_description
3951416 912f79254b8f234bac78c584b3c9e9b2 10.31.21.77 52.11.193.95 ec2amaz-priceapp1 ec2-52-11-193-95.ap-southeast-1.compute.amazonaws.com ec2-52-11-193-95 www.pricingapp.domain.com i-0f88f1ee8f16eea41 ip-10-31-21-77.ap-southeast-1.compute.internal	ec2amaz-priceapp1	AWS	AWS Account: SD9100039 (Retail Price App - Prod) AWS Instance name: ec2amaz-priceapp1 AWS EC2 virtual machine instance id: i-0f88f1ee8f16eea41 AWS Public IP: 52.11.193.95

splunk> .conf21

Asset inventory search examples

Use-case: hunting for ‘solarwinds’ software

Search by software: assets become ‘searchable’ by their installed software.

Real search: this is a real search we ran when Solarwinds supply-chain attack came up. 🧐

```
1  `asset` solarwinds OR (solar winds)
2  | table asset_key asset_installed_software asset_owner_key
```

asset_key	asset_installed_software	asset_owner_key
5106419 | 2a2b4fb9be744cb5b23c3e8e35e0a12a | 10.25.14.72 | ats-mor.addomain.local	solarwinds network automation manager | solarwinds database performance analyzer integration module | solarwinds orion quality of experience | solarwinds orion log viewer | solarwinds log analyzer | solarwinds user device tracker | solarwinds virtual infrastructure monitor | solarwinds voip and network quality manager | solarwinds server & application monitor | solarwinds orion network configuration manager | solarwinds orion netflow traffic analyzer | solarwinds netpath | solarwinds administration service | solarwinds active diagnostics | solarwinds ip address manager | solarwinds orion core services | solarwinds information service | solarwinds agent | solarwinds orion network performance monitor | solarwinds rabbitmq server | nmap npcap | solarwinds orion syslog/traps | solarwinds scp server | oracle open java development kit | microsoft monitoring agent | microsoft windows server | microsoft office access database engine | solarwinds orion network atlas | microsoft azure active directory authentication library for sql server | winpcap team winpcap | solarwinds tftp server | microsoft internet explorer | microsoft sql server management objects | microsoft sql server system clr types | microsoft windows media player | solarwinds job engine | microsoft sql server native client	alice@secret.com | bob@secret.com

Asset inventory search examples

Use-case: CVE-2020-1350 (SIGRed) DNS TCP RCE vulnerability (CVSS: 10)

```
1  `asset` asset_exposure=internet tcp/53 windows server
2  | table asset_key a*ports a*operating_system a*owner_key
```

asset_key	asset_available_ports	asset_operating_system	asset_owner_key
5250671 | ddns01.domain.com	tcp/53 | tcp/80 | tcp/443 | tcp/853 | udp/53	Windows Server 2019	alice@secret.com

Asset inventory search examples

Use-case: compromised account, where **else** is this account seen? (reverse search)
(e.g. pass-the-hash, ransomware/lateral movement attempt)

```
1 `asset` daniel.ferreira
2 | table asset_key asset_owner_key asset_last_logged_on
3 | `recover-mv`
```

asset_key	asset_owner_key	asset_last_logged_on
010b1dcc3877404a8bccf37e948d4aba 5840891 azrsrv1415 10.21.11.18 azrsrv1415.addomain.local	daniel.ferreira@secret.com	bloodhound prv.daniel.ferreira

Asset inventory search examples

Use-case: estate hygiene, asset at risk, missing EDR tool

```
`asset` domain controllers missing-EDR
| table asset_key asset_ad_distinguished_name asset_is_edr*
```

```
`asset` TERM(asset_exposure=extranet) TERM(asset_is_business_critical=true) TERM(asset_is_edr_in_scope=true) TERM(asset_is_edr_seen=false)
| table asset_key asset_owner_key asset_last_seen asset_installed_agent
| `recover-mv`
```

TERM() is your friend: note that if you don't use the TERM() directive in this search, you'll effectively search for 'extranet', 'true' and 'false' LISPY terms only.

Not using **TERM()**:

```
[ AND extranet false index::summary source::asset true ]
```

Using **TERM()**:

```
[ AND asset_exposure=extranet asset_is_business_critical=true asset_is_edr_in_scope=true
asset_is_edr_seen=false index::summary source::asset ]
```

Without **TERM()**, search will return same result but will be **considerable slower!** 😱

Takeaways

- Main generation command for asset inventory must be **|tstats**: allows to handle millions of records.
  - Starting 8.1: command limitation of ~1500 commands per query. Use comma-separated **|eval** for creating fields.
- Data quality is key: ensure a perfect '**host**' field across all data sources, otherwise leverage **ConvertTo-DotNotation** cmdlet (or other method) to add 'key' fields as **PREFIX()**-ready fields.
- Push assets from data sources 'seen/scanned' as of 'yesterday':
  - E.g. Today at 00:00, pull AD Computers with a 'lastlogontimestamp' updated as of the last -24h. AD will refresh this every 14 days.
  - E.g. Today at 00:00, pull assets scanned/discovered by the scanners as of the last -24h. Ensure you scan whole estate every week.
- Before merging different asset inventories, convert all IP addresses to hostname.
- Rely as much as possible on 'machine-generated data'.
  - EDR, AVs, any other agent you may have. Same approach: no 'full-dump' load, instead an incremental one based on 'last seen'. Assets will decay themselves.
- Don't rely on CMDB data unless it comes with a reliable machine-generated 'last seen alive' property.
- Enrich your asset inventory as much as possible. Each field you add will become searchable! 😃
  - Splunk's best value is the ability to **NOT** need field=value searches. Make your field values able to be searched this way.
- Write your asset inventory daily both to Summary Index and KV lookup. Consider to feed Splunk ES with this.
  - But always avoid querying your asset inventory with **|inputlookup**!
- Think very well the values of certain fields: make sure you add relevant 'keywords' to them.
  - E.g. "Azure VM", "AWS load balancer", "Detected Internet facing web server hosted in Azure, port tcp/443 exposed" → `asset` web server azure internet exposed.
- The values of your fields will become '**TERM()** and **PREFIX()**-able' if you don't use minor breakers.
  - Very useful for *_is_* fields!

👋 Help!

Please upvote these on Splunk Ideas:

- Add "OUTPUTMERGE" option to **|lookup**.
- **|collect** command to not double quote fields with just minor breakers.

splunk> .conf21

Thank You

Please provide feedback via the

SESSION SURVEY

splunk> .conf21