Speeding Up Incident Response Using Splunk Indexing Enterprise Metadata

Halvar Myrmo

Telenor

Pål Mathisen

Sopra Steria

.conf2016

splunk>

Disclaimer

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 the 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.

We Will:

Demonstrate how to **simplify** and **speed** up incident response by continually indexing **enterprise metadata** in Splunk.

About

Halvar Myrmo

Senior Security Engineer

Telenor CERT

Previously:

Norwegian Defence

Contact:

halvar.myrmo@telenor.com

Pål Mathisen

Senior Solutions Architect

Sopra Steria

Previously:

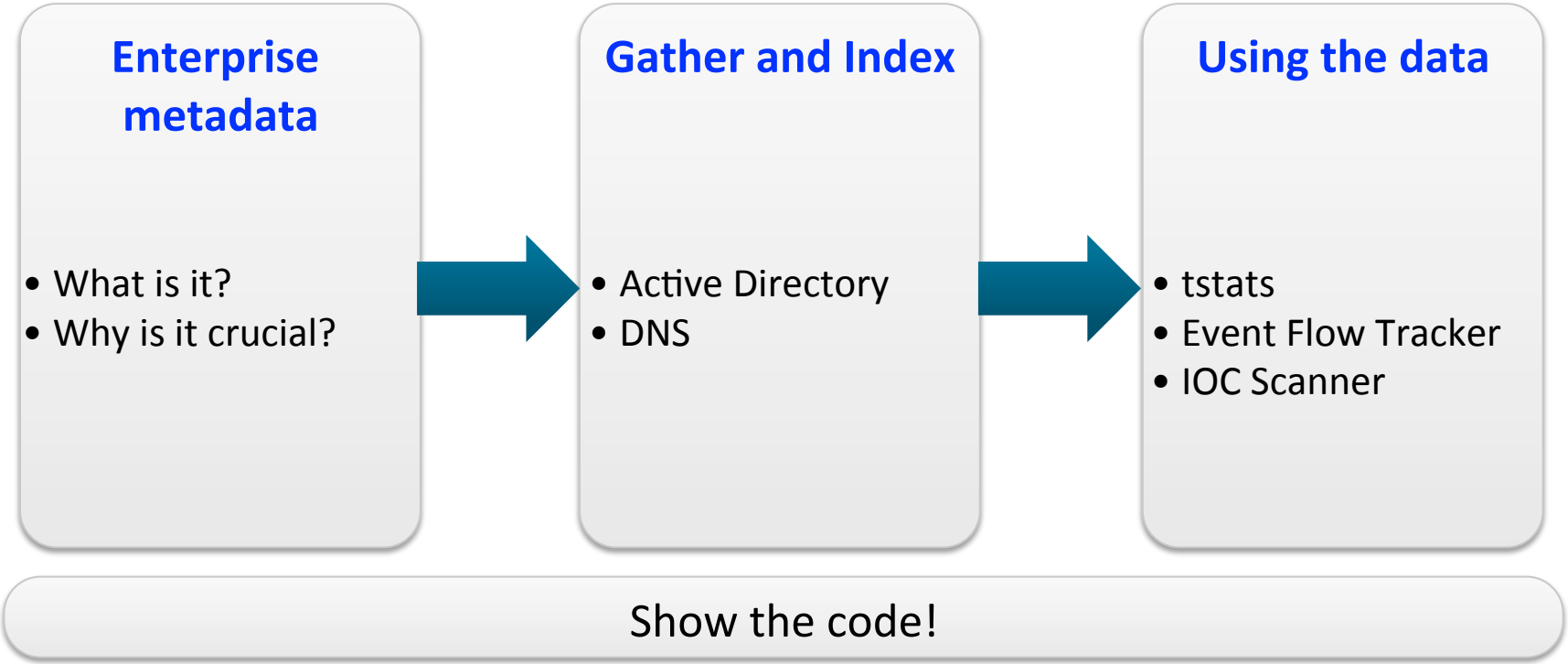
Telenor

Norwegian Defence

Contact:

pal.mathisen@soprasteria.com

Agenda



What Is Enterprise Metadata?

Normal data

- Server logs
- Endpoint logs
- Application logs
- Network logs
- Security logs

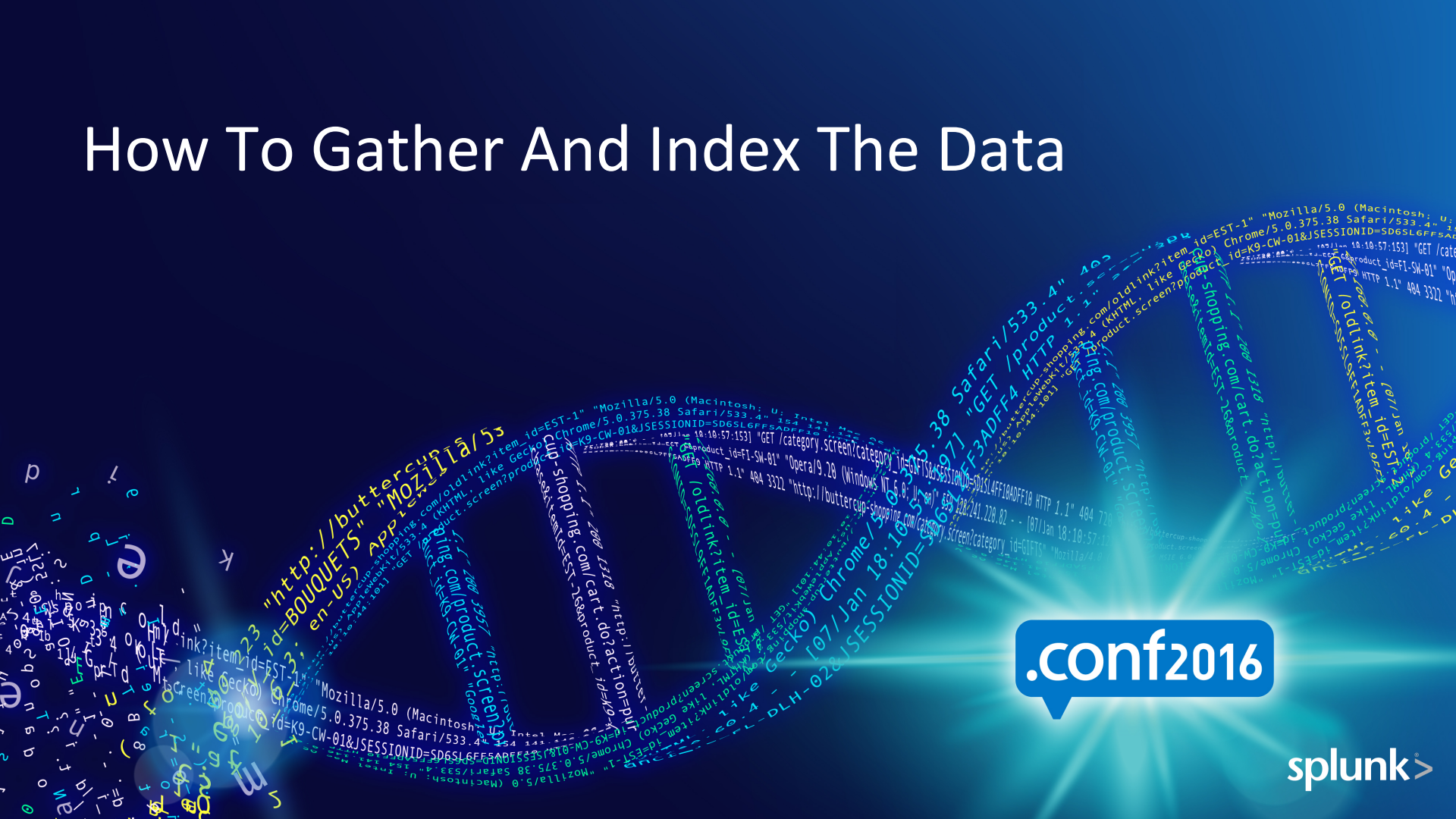
Enterprise metadata

- Active Directory / LDAP
- DNS zones
- Human Resources
- Asset Management
- Network configurations
- VIP lists
- DHCP logs
- Firewall configurations
- Patch status
- Vulnerability scans

Why Is Enterprise Metadata Crucial?

- Enterprises changes over time
- Incidents can last for months or years
- More precise Notable Event enrichment
- Enables historic event enrichment

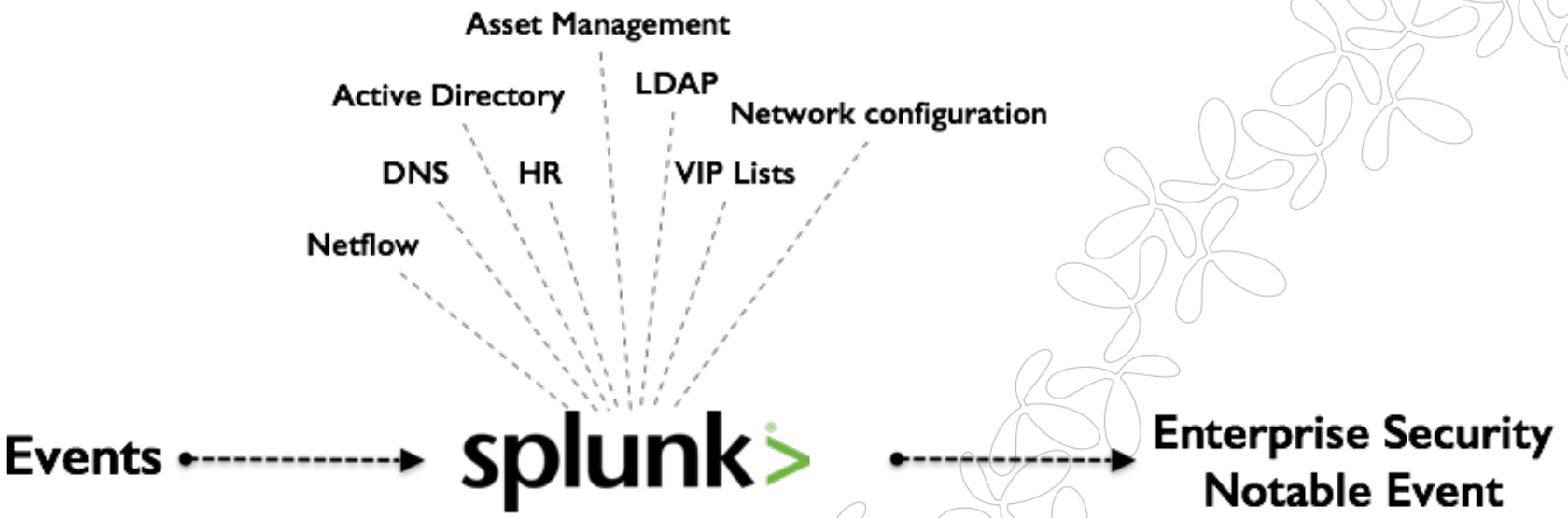
How To Gather And Index The Data



Collect Metadata

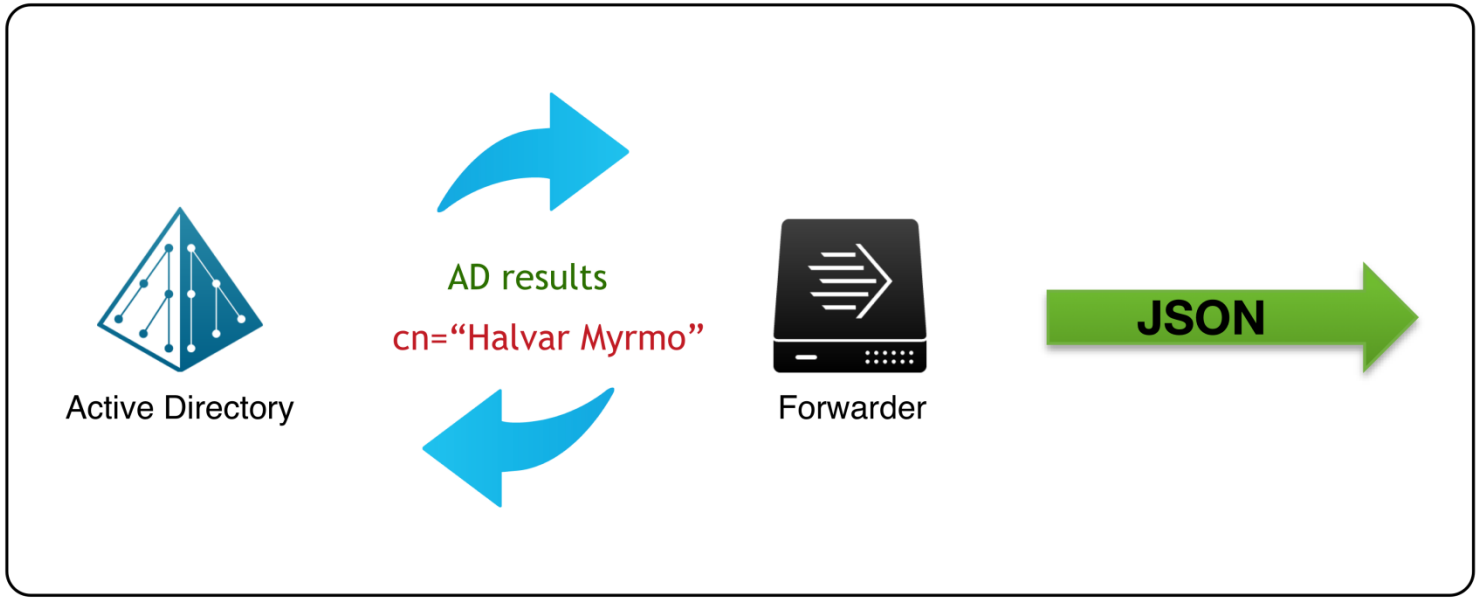
We need to be expert digital cartographers, able to rapidly and correctly map out and discover the nodes and relationships in the part of the enterprise we are investigating.

Gather And Index Metadata



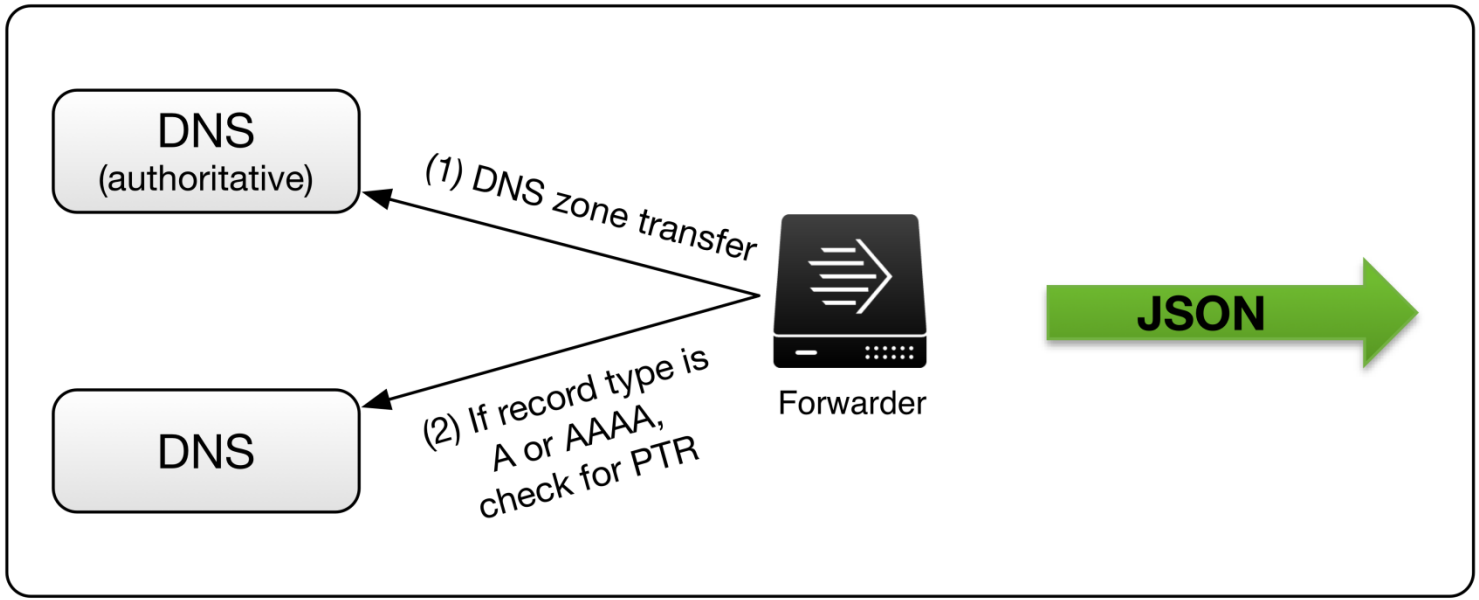
Collection Example: Active Directory

LDAPProcessor.py



Collection Example: DNS Zones

DNSProcessor.py



Engaging Warp Drive

JSON

```
{
    "domain": “example.com",
    "extractTime": "1466053221.0",
     "fqdn": “server1.example.com",
     "name": "server1",
     "rdatasets": [
         {
             "class": "IN",
             "rdata": [
                 {
                     "address": “10.0.0.1”,
```

props.conf

```
[identity:dns:json]
BREAK_ONLY_BEFORE=^{
CHARSET = UTF-8
DATETIME_CONFIG =
INDEXED_EXTRACTIONS = JSON
KV_MODE = none
NO_BINARY_CHECK = true
TIMESTAMP_FIELDS = extractTime
TIME_FORMAT = %s
TRUNCATE = 0
```

How To Use It

Introduction To tstats Command

Counting 100 million events

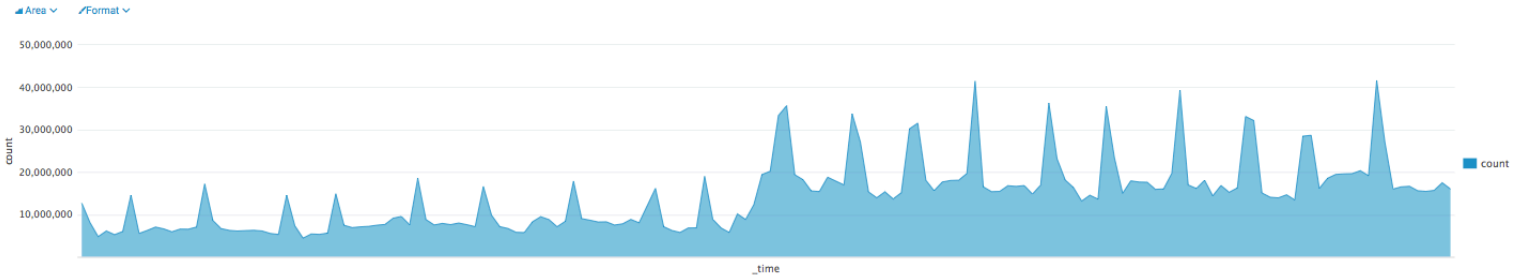
- Normal:
  index=main |stats count
  (42 seconds)

- Warp speed:
  |tstats count WHERE index=main
  (<1 second)

tstats by _*time*

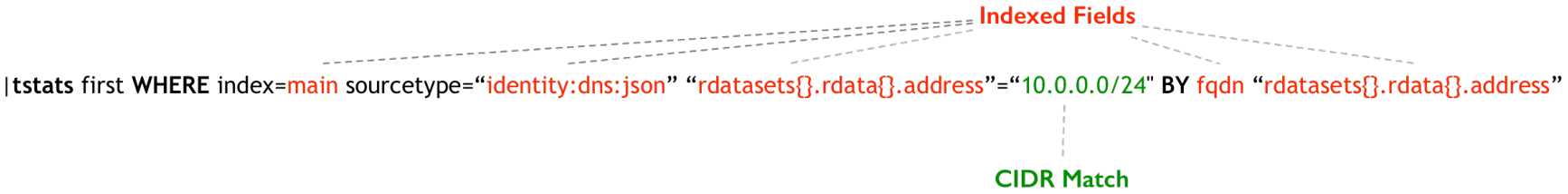
- |tstats count WHERE index=main earliest=-7d@d latest=@d BY _time span=1h
  168 results by scanning 3,011,776,384 events in 4.183 seconds.

- |tstats **prestats=true** count WHERE index=main earliest=-7d@d latest=@d BY _time span=1h |timechart span=1h count
  168 results by scanning 3,011,775,080 events in 4.313 seconds.



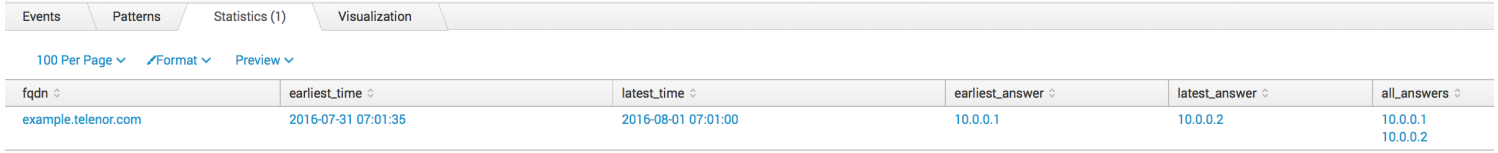
Using tstats On Other Fields

- INDEXED_EXTRACTIONS (props.conf)
  - Logs with headers: (Microsoft IIS, BRO IDS, …)
  - Structured data: (JSON, CSV, PSV, TSB, W3C)



Using tstats On DNS Zone Data

Have any FQDN A-records changed IP-address?




```
| tstats earliest(_time) AS earliest_time latest(_time)
AS latest_time earliest(rdatasets{}.rdata{}.address) as
earliest_answer latest(rdatasets{}.rdata{}.address) AS
latest_answer values(rdatasets{}.rdata{}.address) AS
all_answers WHERE index=sec_ident
sourcetype="tcert:identity:dns:json"
rdatasets{}.type=A earliest=-48h  BY fqdn
| eval earliest_time=strftime(earliest_time, "%F %T")
| eval latest_time=strftime(latest_time, "%F %T")
| WHERE earliest_answer!=latest_answer
```

(≈0.5 second)

vs

```
index=sec_ident sourcetype="tcert:identity:dns:json"
rdatasets{}.type=A earliest=-48h
| stats earliest(_time) AS earliest_time latest(_time)
AS latest_time earliest(rdatasets{}.rdata{}.address) as
earliest_answer latest(rdatasets{}.rdata{}.address) AS
latest_answer values(rdatasets{}.rdata{}.address) AS
all_answers BY fqdn
| WHERE earliest_answer!=latest_answer
| eval earliest_time=strftime(earliest_time, "%F %T")
| eval latest_time=strftime(latest_time, "%F %T")
```

(≈3 seconds)

Using tstats On AD Data

The *User-Account-Control* attribute in AD contains flags that control the behavior of the user account.

Example values:

Hex	Decimal	Description
0x00000020	32	UF_PASSWD_NOTREQD
0x00000200	512	UF_NORMAL_ACCOUNT
0x00010000	65536	UF_DONT_EXPIRE_PASSWD
0x00800000	8388608	UF_PASSWORD_EXPIRED

Example combinations:

Flag combination	Description
544	Normal accounts not subject to password policy
66048	Normal accounts with password that never expire
66080	Normal account with password that never expire, not subject to password policy

Using tstats On AD Data

Normal account with password that never expire, not subject to password policy

```
| tstats first WHERE earliest=-1d
index=sec_ident
sourcetype="tcert:identity:ad:json"
"userAccountControl{}"=66080
BY "sAMAccountName{}", "cn{}"
```

(≈0.5 second)

vs

```
index=sec_ident
sourcetype="tcert:identity:ad:json"
“userAccountControl{}"=66080
| table "sAMAccountName{}", "cn{}"
```

(≈3 seconds)

Using tstats On AD Data

Timeline of when a user gained new memberships in Active Directory

```
| tstats min(_time) AS min WHERE
 earliest=0 latest=now
 index=sec_ident sourcetype="tcert:identity:ad:json"
 "cn{}"="Myrmo Halvar" BY memberOf{}
| eval pretty_time=strftime(min, "%Y-%m-%d")
| table pretty_time, "memberOf{}"
| rename
 pretty_time AS "Added to group"
 "memberOf{}" AS "Group"
```

(≈0.5 second)

vs

```
index=sec_ident sourcetype="tcert:identity:ad:json"
 earliest=0 latest=now
 "cn{}"="Myrmo Halvar"
| stats earliest(_time) as min by memberOf{}
| eval pretty_time=strftime(min, "%Y-%m-%d")
| table pretty_time, "memberOf{}"
| rename pretty_time AS "Added to group"
 "memberOf{}" AS "Group"
```

(≈90 seconds)

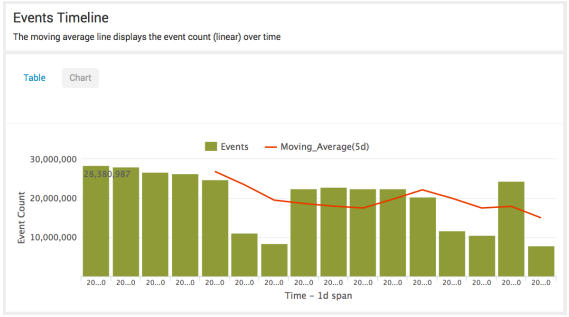
Do you really know where your data is?

Building an Event Flow Tracker

Do You Know If The Logs Arrived?

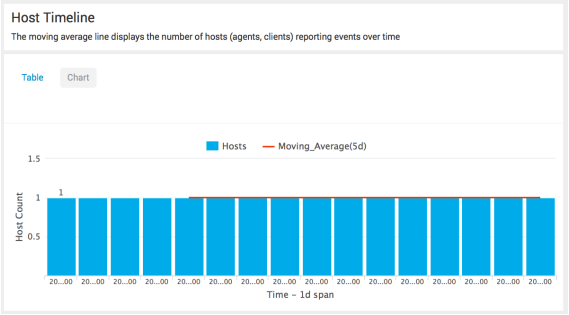
- Count Events

```
| tstats count as Events WHERE index=main earliest=-15d@d latest=now
  BY _time span=1d
| trendline sma5(Events) AS sma
| rename sma AS “Moving_Average(5d)”
```



- Count Hosts

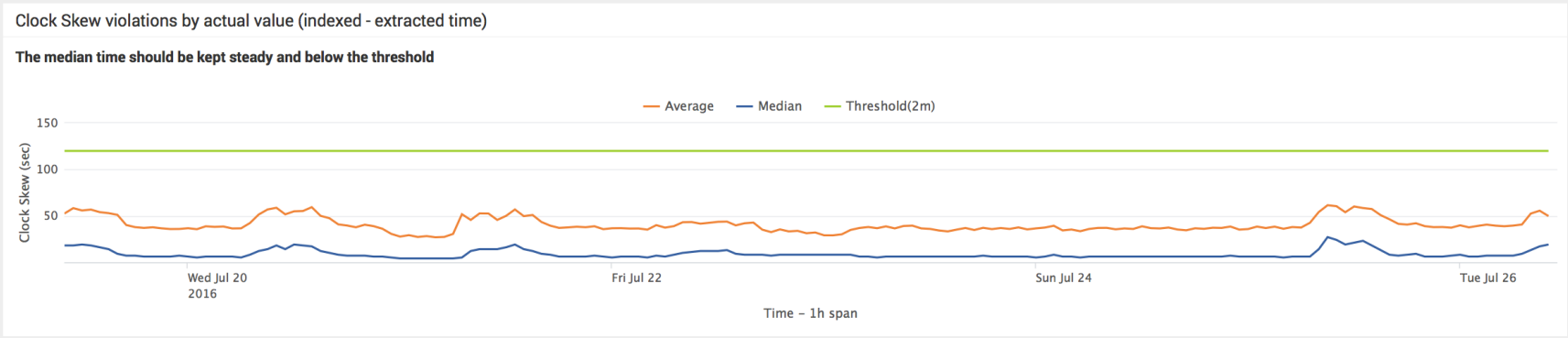
```
| tstats dc(host) AS Hosts WHERE index=main earliest=-15d@d latest=now
  BY _time span=1d
| trendline sma5(Hosts) AS sma
| rename sma AS "Moving_Average(5d)"
```



When Did They Arrive?

- Difference between _time and _indextime

```
| tstats count WHERE index=main sourcetype=syslog BY _indextime, _time span=1s
  | eval diff=_indextime-_time
  | fields - count
  | timechart span=1h cont=t avg(diff) AS "Average", median(diff) As "Median"
  | eval th=120
  | rename th AS "Threshold(2m)"
```



Missing Anything?

- Hosts stopped reporting to this index?

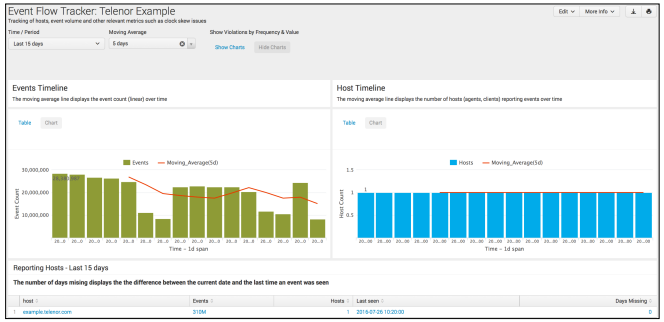
```
| tstats count AS Events WHERE index=important earliest=-15d latest=now BY host, _time
| rex field=host "(?<domain>[^\\\]+)"
| stats sum(Events) as Events, dc(host) AS Hosts, max(_time) as _time by domain
| eval miss = round((now() - _time)/86400)
| convert ctime(_time) timeformat="%Y-%m-%d %H:%M:%S" | sort - Events | eval Events=case(Events>1000000000000, "More than a Quadrillion! \m/",
Events>1000000000, round(Events/1000000000)."T", Events>1000000000, round(Events/1000000000)."B", Events>1000000, round(Events/1000000)."M",
Events>1000, round(Events/1000)."K", 1=1, Events)
| rename _time -> "Last seen", domain -> "Domain/Host", miss AS "Days Missing"
```

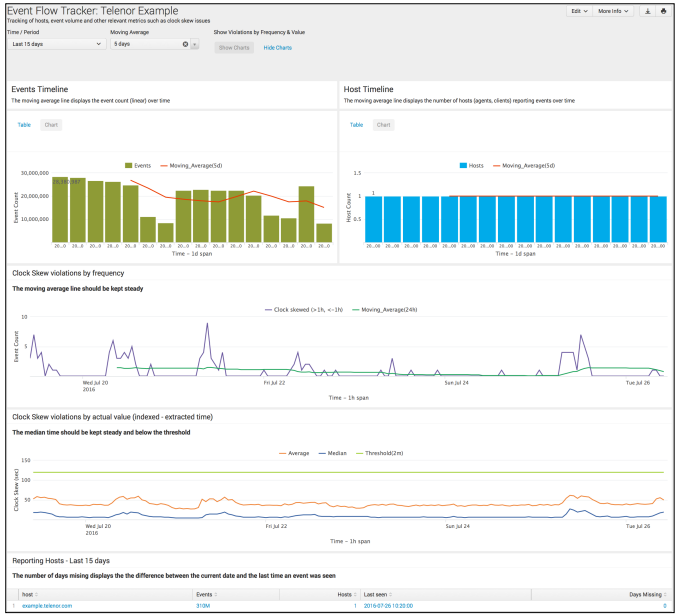
Reporting Hosts - Last 15 days

The number of days mising displays the the difference between the current date and the last time an event was seen

	host	Events	Hosts	Last seen	Days Missing
1	example.telenor.com	310M	1	2016-07-26 10:20:00	0

Splunk Event Flow Tracker



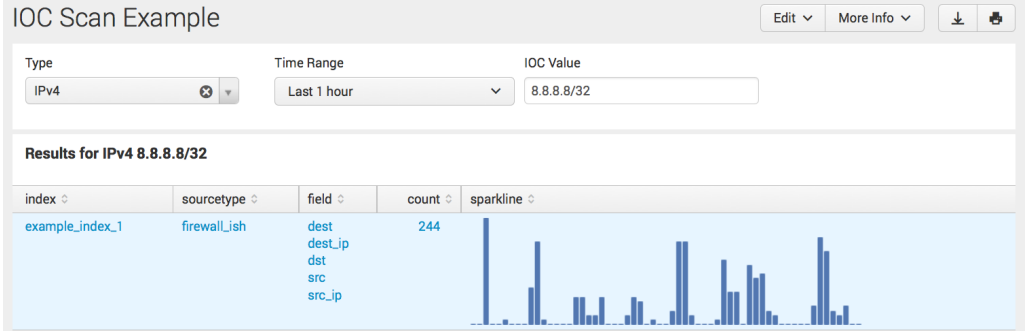


- Event/Host Counter
- Time difference
- Missing Hosts

What can you do with your own indexed fields and tstats?

IOC Scan

- Scheduled saved searches looking for fields containing IPs and Hashes
- Dashbord searching exact *indexes*, *sourcetypes* and *fields*
- Drilldown!



Drilldown:
index=example_index_1 sourcetype=firewall_ish (dest=8.8.8.8/32 OR dest_ip=8.8.8.8/32 OR dst=8.8.8.8/32 OR src=8.8.8.8/32 OR src_ip=8.8.8.8/32) | highlight 8.8.8.8/32

Searches and Dashboard at github.

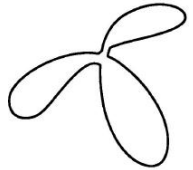
Recap

- Collect and index enterprise metadata daily
- Enrich events with enterprise metadata
- Use tstats to achieve warp speed
- Enable faster and more precise incident response

Show The Code!

- Active Directory / LDAP
- DNS zones
- Event Flow Tracker Dashboard
- IOC Scan Dashboard and Saved Searches

- https://github.com/telenorcert



THANK YOU

splunk>