🔐 Security Analysis with Spans

> **Series:** SPANS | **Notebook:** 6 of 8 | **Created:** December 2025

Protecting Distributed Traces and Ensuring Compliance

This notebook demonstrates how to use span data for security analysis, audit for sensitive data exposure, and ensure compliance with regulations.

Table of Contents

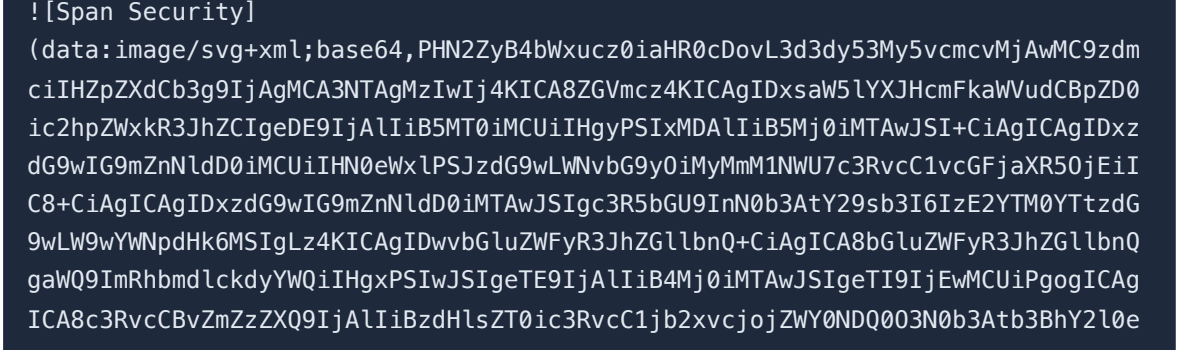
Prerequisites

Before starting this notebook, ensure you have:

- ✅ Completed previous SPANS notebooks (01-05)
- ✅ Understanding of HTTP status codes and security concepts
- ✅ Access to span data containing HTTP attributes
- ✅ Familiarity with OpenPipeline basics

1. Understanding Sensitive Data in Spans

Distributed traces can inadvertently capture sensitive information:



ToxIiAvPgogICAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNkYzI2Mj
Y7c3RvcC1vcGFjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckdyYWR
pZW50IGlkPSJ3YXJuR3JhZCIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMTAwJSI+CiAg
ICAgIDxzdG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNmOTczMTY7c3RvcC1vcGFja
XR5OjEiIC8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6I2VhNT
gwYztzdG9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8bGluZWFyR3J
hZGllbnQgaWQ9InBpcGVsaW5lR3JhZCIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMTAw
JSI+CiAgICAgIDxzdG9wIG9mZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiM4YjVjZjY7c3Rvc
C1vcGFjaXR5OjEiIC8+CiAgICAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3
I6IzdjM2FlZDtzdG9wLW9wYWNpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8Zml
sdGVyIGlkPSJzZWNTaGFkb3ciPgogICAgICA8ZmVEcm9wU2hhZG93IGR4PSIyIiBkeT0iMiIgc3Rk
RGV2aWF0aW9uPSIzIiBmbG9vZC1vcGFjaXR5PSIwLjE1Ii8+CiAgICA8L2ZpbHRlcj4KICAgIDxtY
XJrZXIgaWQ9InNlY0Fycm93IiBtYXJrZXJXaWR0aD0iOCIgbWFya2VySGVpZ2h0PSI2IiByZWZYPS
I3IiByZWZZPSIzIiBvcmllbnQ9ImF1dG8iPgogICAgICA8cG9seWdvbiBwb2ludHM9IjAgMCwgOCA
zLCAwIDYiIGZpbGw9IiM2NDc0OGIiLz4KICAgIDwvbWFya2VyPgogIDwvZGVmcz4KCiAgPCEtLSBC
YWNrZ3JvdW5kIC0tPgogIDxyZWN0IHdpZHRoPSI3NTAiIGhlaWdodD0iMzIwIiBmaWxsPSIjZjhmO
WZhIiByeD0iMTAiLz4KCiAgPCEtLSBUaXRsZSAtLT4KICA8dGV4dCB4PSIzNzUiIHk9IjI4IiBmb2
50LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTgiIGZvbnQtd2VpZ2h0PSJ
ib2xkIiBmaWxsPSIjMzMzIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5TcGFuIFNlY3VyaXR5OiBQcm90
ZWN0aW5nIFNlbnNpdGl2ZSBEYXRhPC90ZXh0PgoKICA8IS0tIFBJSSBSaXNrcyBTZWN0aW9uIC0tP
gogIDxyZWN0IHg9IjQwIiB5PSI1MCIgd2lkdGg9IjIwMCIgaGVpZ2h0PSIxODAiIHJ4PSI4IiBmaW
xsPSIjZmZmIiBzdHJva2U9IiNmZWNhY2EiIHN0cm9rZS13aWR0aD0iMiIgZmlsdGVyPSJ1cmwoI3N
lY1NoYWRvdykiLz4KICA8cmVjdCB4PSI0MCIgeT0iNTAiIHdpZHRoPSIyMDAiIGhlaWdodD0iMzUi
IHJ4PSI4IiBmaWxsPSJ1cmwoI2RhbmdlckdyYWQpIi8+CiAgPHJlY3QgeD0iNDAiIHk9Ijc1IiB3a
WR0aD0iMjAwIiBoZWlnaHQ9IjEwIiBmaWxsPSJ1cmwoI2RhbmdlckdyYWQpIi8+CiAgPHRleHQgeD
0iMTQwIiB5PSI3NSIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjE
zIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2hpdGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPlBJ
SSBpbiBTcGFuczwvdGV4dD4KCiAgPHRleHQgeD0iNTUiIHk9IjEwNSIgZm9udC1mYW1pbHk9IkFya
WFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmaWxsPSIjMzMzIj5Db21tb24gRXhwb3N1cm
UgUG9pbnRzOjwvdGV4dD4KICA8dGV4dCB4PSI1NSIgeT0iMTI1IiBmb250LWZhbWlseT0iQXJpYWw
sIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NjYiPi0gaHR0cC51cmwgd2l0aCBx
dWVyeSBwYXJhbXM8L3RleHQ+CiAgPHRleHQgeD0iNTUiIHk9IjE0MiIgZm9udC1mYW1pbHk9IkFya
WFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjY2Ij4tIGRiLnN0YXRlbWVudC
B3aXRoIHZhbHVlczwvdGV4dD4KICA8dGV4dCB4PSI1NSIgeT0iMTU5IiBmb250LWZhbWlseT0iQXJ
pYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NjYiPi0gc3Bhbi5uYW1lIHdp
dGggdXNlciBJRHM8L3RleHQ+CiAgPHRleHQgeD0iNTUiIHk9IjE3NiIgZm9udC1mYW1pbHk9IkFya
WFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjY2Ij4tIEN1c3RvbSBhdHRyaW
J1dGVzPC90ZXh0PgogIDx0ZXh0IHg9IjU1IiB5PSIxOTYiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2F
ucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzY2NiI+LSBFcnJvciBtZXNzYWdlczwvdGV4
dD4KICA8dGV4dCB4PSI1NSIgeT0iMjE2IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiI
GZvbnQtc2l6ZT0iMTAiIGZpbGw9IiNlZjQ0NDQiIGZvbnQtd2VpZ2h0PSJib2xkIj5BdWRpdCByZW
d1bGFybHkhPC90ZXh0PgoKICA8IS0tIE9wZW5QaXBlbGluZSBTb2x1dGlvbiAtLT4KICA8cmVjdCB
4PSIyODAiIHk9IjUwIiB3aWR0aD0iMjAwIiBoZWlnaHQ9IjE4MCIgcng9IjgiIGZpbGw9IiNmZmYi
IHN0cm9rZT0iI2M0YjVmZCIgc3Ryb2tlLXdpZHRoPSIyIiBmaWx0ZXI9InVybCgjc2VjU2hhZG93K
SIvPgogIDxyZWN0IHg9IjI4MCIgeT0iNTAiIHdpZHRoPSIyMDAiIGhlaWdodD0iMzUiIHJ4PSI4Ii
BmaWxsPSJ1cmwoI3BpcGVsaW5lR3JhZCkiLz4KICA8cmVjdCB4PSIyODAiIHk9Ijc1IiB3aWR0aD0
iMjAwIiBoZWlnaHQ9IjEwIiBmaWxsPSJ1cmwoI3BpcGVsaW5lR3JhZCkiLz4KICA8dGV4dCB4PSIz
ODAiIHk9Ijc1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTMiI

GZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+T3Blbl
BpcGVsaW5lPC90ZXh0PgoKICA8dGV4dCB4PSIyOTUiIHk9IjEwNSIgZm9udC1mYW1pbHk9IkFyaWF
sLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmaWxsPSIjMzMzIj5EYXRhIFByb2Nlc3Npbmc6
PC90ZXh0PgogIDx0ZXh0IHg9IjI5NSIgeT0iMTI1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc
2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NjYiPi0gTWFzayBzZW5zaXRpdmUgZmllbGRzPC
90ZXh0PgogIDx0ZXh0IHg9IjI5NSIgeT0iMTQyIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2V
yaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NjYiPi0gRHJvcCBzcGVjaWZpYyBhdHRyaWJ1dGVz
PC90ZXh0PgogIDx0ZXh0IHg9IjI5NSIgeT0iMTU5IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc
2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM2NjYiPi0gUmVkYWN0IHBhdHRlcm5zIChTU04sIE
NDKTwvdGV4dD4KICA8dGV4dCB4PSIyOTUiIHk9IjE3NiIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5
zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjY2Ij4tIEhhc2ggaWRlbnRpZmllcnM8L3Rl
eHQ+CiAgPHRleHQgeD0iMjk1IiB5PSIxOTYiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZ
iIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzY2NiI+LSBSb3V0ZSB0byBzZWN1cmUgYnVja2V0czwvdG
V4dD4KICA8dGV4dCB4PSIyOTUiIHk9IjIxNiIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcml
mIiBmb250LXNpemU9IjEwIiBmaWxsPSIjOGI1Y2Y2IiBmb250LXdlaWdodD0iYm9sZCI+Q29uZmln
dXJlIGJlZm9yZSBpbmdlc3Rpb248L3RleHQ+CgogIDwhLS0gQWNjZXNzIENvbnRyb2wgLS0+CiAgP
HJlY3QgeD0iNTIwIiB5PSI1MCIgd2lkdGg9IjIwMCIgaGVpZ2h0PSIxODAiIHJ4PSI4IiBmaWxsPS
IjZmZmIiBzdHJva2U9IiNiYmY3ZDAiIHN0cm9rZS13aWR0aD0iMiIgZmlsdGVyPSJ1cmwoI3NlY1N
oYWRvdykiLz4KICA8cmVjdCB4PSI1MjAiIHk9IjUwIiB3aWR0aD0iMjAwIiBoZWlnaHQ9IjM1IiBy
eD0iOCIgZmlsbD0idXJsKCNzaGllbGRHcmFkKSIvPgogIDxyZWN0IHg9IjUyMCIgeT0iNzUiIHdpZ
HRoPSIyMDAiIGhlaWdodD0iMTAiIGZpbGw9InVybCgjc2hpZWxkR3JhZCkiLz4KICA8dGV4dCB4PS
I2MjAiIHk9Ijc1IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTM
iIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+QWNj
ZXNzIENvbnRyb2w8L3RleHQ+CgogIDx0ZXh0IHg9IjUzNSIgeT0iMTA1IiBmb250LWZhbWlseT0iQ
XJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTEiIGZpbGw9IiMzMzMiPlByb3RlY3Rpb24gTG
F5ZXJzOjwvdGV4dD4KICA8dGV4dCB4PSI1MzUiIHk9IjEyNSIgZm9udC1mYW1pbHk9IkFyaWFsLCB
zYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjY2Ij4tIEJ1Y2tldC1sZXZlbCBwZXJt
aXNzaW9uczwvdGV4dD4KICA8dGV4dCB4PSI1MzUiIHk9IjE0MiIgZm9udC1mYW1pbHk9IkFyaWFsL
CBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjY2Ij4tIEZpZWxkLWxldmVsIHJlc3
RyaWN0aW9uczwvdGV4dD4KICA8dGV4dCB4PSI1MzUiIHk9IjE1OSIgZm9udC1mYW1pbHk9IkFyaWF
sLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjNjY2Ij4tIERhdGEgc2VnbWVudHM8
L3RleHQ+CiAgPHRleHQgeD0iNTM1IiB5PSIxNzYiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZ
XJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzY2NiI+LSBJQU0gcG9saWNpZXM8L3RleHQ+CiAgPH
RleHQgeD0iNTM1IiB5PSIxOTYiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1
zaXplPSIxMCIgZmlsbD0iIzY2NiI+LSBBdWRpdCBsb2dnaW5nPC90ZXh0PgogIDx0ZXh0IHg9IjUz
NSIgeT0iMjE2IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiI
GZpbGw9IiMyMmM1NWUiIGZvbnQtd2VpZ2h0PSJib2xkIj5MZWFzdCBwcml2aWxlZ2UgcHJpbmNpcG
xlPC90ZXh0PgoKICA8IS0tIEFycm93cyAtLT4KICA8bGluZSB4MT0iMjQwIiB5MT0iMTQwIiB4Mj0
iMjc1IiB5Mj0iMTQwIiBzdHJva2U9IiM2NDc0OGIiIHN0cm9rZS13aWR0aD0iMiIgbWFya2VyLWVu
ZD0idXJsKCNzZWNBcnJvdykiLz4KICA8bGluZSB4MT0iNDgwIiB5MT0iMTQwIiB4Mj0iNTE1IiB5M
j0iMTQwIiBzdHJva2U9IiM2NDc0OGIiIHN0cm9rZS13aWR0aD0iMiIgbWFya2VyLWVuZD0idXJsKC
NzZWNBcnJvdykiLz4KCiAgPCEtLSBCb3R0b20gYXVkaXQgcXVlcnkgZXhhbXBsZSAtLT4KICA8cmV
jdCB4PSI0MCIgeT0iMjUwIiB3aWR0aD0iNjgwIiBoZWlnaHQ9IjU1IiByeD0iNiIgZmlsbD0iIzFl
MjkzYiIvPgogIDx0ZXh0IHg9IjU1IiB5PSIyNzIiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZ
XJpZiIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IiM5NGEzYjgiPlBJSS
BBdWRpdCBRdWVyeTo8L3RleHQ+CiAgPHRleHQgeD0iMTcwIiB5PSIyNzIiIGZvbnQtZmFtaWx5PSJ
tb25vc3BhY2UiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiMyMmM1NWUiPmZldGNoPC90ZXh0PgogIDx0
ZXh0IHg9IjIwNSIgeT0iMjcyIiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwI

iBmaWxsPSIjZjhmYWZjIj5zcGFuczwvdGV4dD4KICA8dGV4dCB4PSIyNTAiIHk9IjI3MiIgZm9udC
1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0iIzk0YTNiOCI+fDwvdGV4dD4
KICA8dGV4dCB4PSIyNjAiIHk9IjI3MiIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXpl
PSIxMCIgZmlsbD0iI2Y5NzMxNiI+ZmlsdGVyPC90ZXh0PgogIDx0ZXh0IHg9IjMwMiIgeT0iMjcyI
iBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjZjhmYWZjIj5tYX
RjaGVzUGhyYXNlKGh0dHAudXJsLDwvdGV4dD4KICA8dGV4dCB4PSI0NTUiIHk9IjI3MiIgZm9udC1
mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0iI2ZiYmYyNCI+IiplbWFpbD0q
IjwvdGV4dD4KICA8dGV4dCB4PSI1MjAiIHk9IjI3MiIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZ
m9udC1zaXplPSIxMCIgZmlsbD0iI2Y4ZmFmYyI+KTwvdGV4dD4KICA8dGV4dCB4PSI1NSIgeT0iMj
kyIiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjOTRhM2I4Ij4
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgIG9yIG1hdGNoZXNQaHJhc2UoaHR0cC51
cmwsICIqcGFzc3dvcmQ9KiIpIG9yIG1hdGNoZXNQaHJhc2UoaHR0cC51cmwsICIqdG9rZW49KiIpP
C90ZXh0Pgo8L3N2Zz4K)

Fields to Consider for Protection

Field	Risk	Recommendation
`url.path`	PII in path/params	Mask or use http.route
`http.request.header.*`	Auth tokens/cookies	Drop sensitive headers
`db.statement`	SQL with user data	Parameterize or mask
`exception.stacktrace`	Variable values	Truncate or filter
`messaging.payload`	Message content	Drop or encrypt
Custom attributes	Business data	Evaluate case by case

---

2. PII Audit Queries

Regularly audit your span data for potential PII exposure. These queries help identify data that may need masking or removal.

Step 1: Find URLs with Email Addresses

```dql
// Find URLs that might contain email addresses
// Look for @ symbol or URL-encoded %40
fetch spans
| filter isNotNull(url.path)
| filter contains(url.path, "@") or contains(url.path, "%40")
| fields start_time,
         dt.entity.service,
         http.route,
         url.path
| dedup http.route
| limit 20
```

```

### Step 2: Find URLs with User Identifiers

```dql
// Find URLs with potential user identifiers
fetch spans
| filter isNotNull(url.path)
| filter contains(url.path, "user") or
        contains(url.path, "email") or
        contains(url.path, "customer") or
        contains(url.path, "account")
| fields dt.entity.service,
         http.route,
         url.path
| dedup http.route
| limit 20
```

Step 3: Find Sensitive Query Parameters

```dql
// Find URLs with sensitive query parameters
// These suggest credentials or tokens in URLs
fetch spans
| filter isNotNull(url.path)
| filter contains(url.path, "password") or
        contains(url.path, "token") or
        contains(url.path, "key") or
        contains(url.path, "secret") or
        contains(url.path, "auth")
| fields dt.entity.service,
         http.route,
         url.path
| dedup http.route
| limit 20
```

Step 4: Audit Database Queries for PII

```dql
// Find database queries that might contain sensitive data
fetch spans
| filter isNotNull(db.statement)
| filter contains(db.statement, "password") or
        contains(db.statement, "ssn") or
        contains(db.statement, "credit") or
        contains(db.statement, "email") or
```
```

```
        contains(db.statement, "phone")
| fields dt.entity.service,
         db.system,
         db.name,
         db.statement
| limit 20
```

Step 5: Check Error Messages for Data Leakage

```dql
// Audit error messages for potential data leakage
fetch spans
| filter span.status_code == "error"
| filter isNotNull(span.status_message)
| fields dt.entity.service,
         span.name,
         span.status_message
| limit 20
```

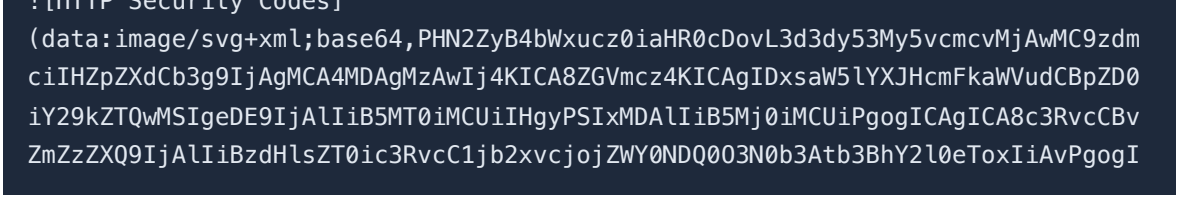
Summary: Which Fields Contain Potentially Sensitive Data?

```dql
// Identify which potentially sensitive fields are present
fetch spans
| summarize {
    total_spans = count(),
    has_url = countIf(isNotNull(url.path)),
    has_db_statement = countIf(isNotNull(db.statement)),
    has_exception = countIf(isNotNull(exception.stacktrace))
  }
| fieldsAdd url_percent = (has_url * 100.0) / total_spans
| fieldsAdd db_percent = (has_db_statement * 100.0) / total_spans
| fieldsAdd exception_percent = (has_exception * 100.0) / total_spans
```

---

3. HTTP Status Code Security Analysis

HTTP status codes can reveal security-relevant patterns:



```
```

CAgICA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNkYzI2MjY7c3RvcC1vcG
FjaXR5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckdyYWRpZW50IGlkPSJ
jb2RlNDAzIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIwJSI+CiAgICAgIDxzdG9wIG9m
ZnNldD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiNmOTczMTY7c3RvcC1vcGFjaXR5OjEiIC8+CiAgI
CAgIDxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6I2VhNTgwYztzdG9wLW9wYW
NpdHk6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8bGluZWFyR3JhZGllbnQgaWQ9ImN
vZGU0MDQiIHgxPSIwJSIgeTE9IjAlIiB4Mj0iMTAwJSIgeTI9IjAlIj4KICAgICAgPHN0b3Agb2Zm
c2V0PSIwJSIgc3R5bGU9InN0b3AtY29sb3I6I2VhYjMwODtzdG9wLW9wYWNpdHk6MSIgLz4KICAgI
CAgPHN0b3Agb2Zmc2V0PSIxMDAlIiBzdHlsZT0ic3RvcC1jb2xvcjojY2E4YTA0O3N0b3Atb3BhY2
l0eToxIiAvPgogICAgPC9saW5lYXJHcmFkaWVudD4KICAgIDxsaW5lYXJHcmFkaWVudCBpZD0iY29
kZTQyOSIgeDE9IjAlIiB5MT0iMCUiIHgyPSIxMDAlIiB5Mj0iMCUiPgogICAgICA8c3RvcCBvZmZz
ZXQ9IjAlIiBzdHlsZT0ic3RvcC1jb2xvcjojOGI1Y2Y2O3N0b3Atb3BhY2l0eToxIiAvPgogICAgI
CA8c3RvcCBvZmZzZXQ9IjEwMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiM3YzNhZWQ7c3RvcC1vcGFjaX
R5OjEiIC8+CiAgICA8L2xpbmVhckdyYWRpZW50PgogICAgPGxpbmVhckdyYWRpZW50IGlkPSJjb2R
lNTAwIiB4MT0iMCUiIHkxPSIwJSIgeDI9IjEwMCUiIHkyPSIwJSI+CiAgICAgIDxzdG9wIG9mZnNl
dD0iMCUiIHN0eWxlPSJzdG9wLWNvbG9yOiMxNDk2ZmY7c3RvcC1vcGFjaXR5OjEiIC8+CiAgICAgI
DxzdG9wIG9mZnNldD0iMTAwJSIgc3R5bGU9InN0b3AtY29sb3I6IzBhNjRiYztzdG9wLW9wYWNpdH
k6MSIgLz4KICAgIDwvbGluZWFyR3JhZGllbnQ+CiAgICA8ZmlsdGVyIGlkPSJjb2RlU2hhZG93Ij4
KICAgICAgPGZlRHJvcFNoYWRvdyBkeD0iMiIgZHk9IjIiIHN0ZERldmlhdGlvbj0iMiIgZmxvb2Qt
b3BhY2l0eT0iMC4xNSIvPgogICAgPC9maWx0ZXI+CiAgPC9kZWZzPgoKICA8IS0tIEJhY2tncm91b
mQgLS0+CiAgPHJlY3Qgd2lkdGg9IjgwMCIgaGVpZ2h0PSIzMDAiIGZpbGw9IiNmOGY5ZmEiIHJ4PS
IxMCIvPgoKICA8IS0tIFRpdGxlIC0tPgogIDx0ZXh0IHg9IjQwMCIgeT0iMjgiIGZvbnQtZmFtaWx
5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxOCIgZm9udC13ZWlnaHQ9ImJvbGQiIGZp
bGw9IiMzMzMiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkhUVFAgU3RhdHVzIENvZGVzIGZvciBTZWN1c
ml0eSBBbmFseXNpczwvdGV4dD4KCiAgPCEtLSA0MDEgVW5hdXRob3JpemVkIC0tPgogIDxyZWN0IH
g9IjMwIiB5PSI1MCIgd2lkdGg9IjE0NSIgaGVpZ2h0PSIxMDAiIHJ4PSI4IiBmaWxsPSJ1cmwoI2N
vZGU0MDEpIiBmaWx0ZXI9InVybCgjY29kZVNoYWRvdykiLz4KICA8dGV4dCB4PSIxMDIiIHk9Ijgw
IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMjgiIGZvbnQtd2VpZ
2h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+NDAxPC90ZXh0PgogID
x0ZXh0IHg9IjEwMiIgeT0iMTAwIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQ
tc2l6ZT0iMTEiIGZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSJ3aGl0ZSIgdGV4dC1hbmNob3I9Im1p
ZGRsZSI+VW5hdXRob3JpemVkPC90ZXh0PgogIDx0ZXh0IHg9IjEwMiIgeT0iMTE4IiBmb250LWZhb
WlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NS
wyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+RmFpbGVkIGF1dGhlbnRpY2F0aW9uPC90ZXh
0PgogIDx0ZXh0IHg9IjEwMiIgeT0iMTMzIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYi
IGZvbnQtc2l6ZT0iMTAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9I
m1pZGRsZSI+SW52YWxpZCBjcmVkZW50aWFsczwvdGV4dD4KCiAgPCEtLSA0MDMgRm9yYmlkZGVuIC
0tPgogIDxyZWN0IHg9IjE4NSIgeT0iNTAiIHdpZHRoPSIxNDUiIGhlaWdodD0iMTAwIiByeD0iOCI
gZmlsbD0idXJsKCNjb2RlNDAzKSIgZmlsdGVyPSJ1cmwoI2NvZGVTaGFkb3cpIi8+CiAgPHRleHQg
eD0iMjU3IiB5PSI4MCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9I
jI4IiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2hpdGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPj
QwMzwvdGV4dD4KICA8dGV4dCB4PSIyNTciIHk9IjEwMCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5
zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2hpdGUiIHRl
eHQtYW5jaG9yPSJtaWRkbGUiPkZvcmJpZGRlbjwvdGV4dD4KICA8dGV4dCB4PSIyNTciIHk9IjExO
CIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2
JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGUiPkFjY2VzcyBjb250cm9sIHZ
pb2xhdGlvbjwvdGV4dD4KICA8dGV4dCB4PSIyNTciIHk9IjEzMyIgZm9udC1mYW1pbHk9IkFyaWFs
LCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiI

HRleHQtYW5jaG9yPSJtaWRkbGUiPkluc3VmZmljaWVudCBwZXJtaXNzaW9uczwvdGV4dD4KCiAgPC
EtLSA0MDQgTm90IEZvdW5kIC0tPgogIDxyZWN0IHg9IjM0MCIgeT0iNTAiIHdpZHRoPSIxNDUiIGh
laWdodD0iMTAwIiByeD0iOCIgZmlsbD0idXJsKCNjb2RlNDA0KSIgZmlsdGVyPSJ1cmwoI2NvZGVT
aGFkb3cpIi8+CiAgPHRleHQgeD0iNDEyIiB5PSI4MCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zL
XNlcmlmIiBmb250LXNpemU9IjI4IiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0id2hpdGUiIHRleH
QtYW5jaG9yPSJtaWRkbGUiPjQwNDwvdGV4dD4KICA8dGV4dCB4PSI0MTIiIHk9IjEwMCIgZm9udC1
mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9s
ZCIgZmlsbD0id2hpdGUiIHRleHQtYW5jaG9yPSJtaWRkbGUiPk5vdCBGb3VuZDwvdGV4dD4KICA8d
GV4dCB4PSI0MTIiIHk9IjExOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LX
NpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9yPSJtaWRkbGU
iPkVuZHBvaW50IHByb2Jpbmc8L3RleHQ+CiAgPHRleHQgeD0iNDEyIiB5PSIxMzMiIGZvbnQtZmFt
aWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1L
DI1NSwwLjkpIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj4oaWYgc3Bpa2luZyk8L3RleHQ+CgogIDwhLS
0gNDI5IFJhdGUgTGltaXRlZCAtLT4KICA8cmVjdCB4PSI0OTUiIHk9IjUwIiB3aWR0aD0iMTQ1IiB
oZWlnaHQ9IjEwMCIgcng9IjgiIGZpbGw9InVybCgjY29kZTQyOSkiIGZpbHRlcj0idXJsKCNjb2Rl
U2hhZG93KSIvPgogIDx0ZXh0IHg9IjU2NyIgeT0iODAiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fuc
y1zZXJpZiIgZm9udC1zaXplPSIyOCIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IndoaXRlIiB0ZX
h0LWFuY2hvcj0ibWlkZGxlIj40Mjk8L3RleHQ+CiAgPHRleHQgeD0iNTY3IiB5PSIxMDAiIGZvbnQ
tZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMSIgZm9udC13ZWlnaHQ9ImJv
bGQiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5Ub28gTWFueSBSZXF1ZXN0czwvd
GV4dD4KICA8dGV4dCB4PSI1NjciIHk9IjExOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcm
lmIiBmb250LXNpemU9IjEwIiBmaWxsPSJyZ2JhKDI1NSwyNTUsMjU1LDAuOSkiIHRleHQtYW5jaG9
yPSJtaWRkbGUiPlJhdGUgbGltaXRpbmcgdHJpZ2dlcmVkPC90ZXh0PgogIDx0ZXh0IHg9IjU2NyIg
eT0iMTMzIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpb
Gw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+UG9zc2libGUgYW
J1c2U8L3RleHQ+CgogIDwhLS0gNTAwIFNlcnZlciBFcnJvciAtLT4KICA8cmVjdCB4PSI2NTAiIHk
9IjUwIiB3aWR0aD0iMTIwIiBoZWlnaHQ9IjEwMCIgcng9IjgiIGZpbGw9InVybCgjY29kZTUwMCki
IGZpbHRlcj0idXJsKCNjb2RlU2hhZG93KSIvPgogIDx0ZXh0IHg9IjcxMCIgeT0iODAiIGZvbnQtZ
mFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIyOCIgZm9udC13ZWlnaHQ9ImJvbG
QiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj41eHg8L3RleHQ+CiAgPHRleHQgeD0
iNzEwIiB5PSIxMDAiIGZvbnQtZmFtaWx5PSJBcmlhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIx
MSIgZm9udC13ZWlnaHQ9ImJvbGQiIGZpbGw9IndoaXRlIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5TZ
XJ2ZXIgRXJyb3I8L3RleHQ+CiAgPHRleHQgeD0iNzEwIiB5PSIxMTgiIGZvbnQtZmFtaWx5PSJBcm
lhbCwgc2Fucy1zZXJpZiIgZm9udC1zaXplPSIxMCIgZmlsbD0icmdiYSgyNTUsMjU1LDI1NSwwLjk
pIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5FeHBsb2l0IGF0dGVtcHRzPC90ZXh0PgogIDx0ZXh0IHg9
IjcxMCIgeT0iMTMzIiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iM
TAiIGZpbGw9InJnYmEoMjU1LDI1NSwyNTUsMC45KSIgdGV4dC1hbmNob3I9Im1pZGRsZSI+U3lzdG
VtIGZhaWx1cmVzPC90ZXh0PgoKICA8IS0tIEFsZXJ0IFBhdHRlcm5zIFNlY3Rpb24gLS0+CiAgPHJ
lY3QgeD0iMzAiIHk9IjE2NSIgd2lkdGg9IjM2MCIgaGVpZ2h0PSIxMjAiIHJ4PSI2IiBmaWxsPSIj
ZmVmMmYyIiBzdHJva2U9IiNmZWNhY2EiIHN0cm9rZS13aWR0aD0iMSIvPgogIDx0ZXh0IHg9IjIxM
CIgeT0iMTg4IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTIiIG
ZvbnQtd2VpZ2h0PSJib2xkIiBmaWxsPSIjOTkxYjFiIiB0ZXh0LWFuY2hvcj0ibWlkZGxlIj5TZWN
1cml0eSBBbGVydCBQYXR0ZXJuczwvdGV4dD4KCiAgPHRleHQgeD0iNTAiIHk9IjIxMCIgZm9udC1m
YW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb250LXNpemU9IjEwIiBmaWxsPSIjOTkxYjFiIj5Tc
GlrZSBpbiA0MDFzIOKGkiBCcnV0ZSBmb3JjZSAvIGNyZWRlbnRpYWwgc3R1ZmZpbmc8L3RleHQ+Ci
AgPHRleHQgeD0iNTAiIHk9IjIyOCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBmb25
0LXNpemU9IjEwIiBmaWxsPSIjOTkxYjFiIj5TcGlrZSBpbiA0MDNzIOKGkiBQcml2aWxlZ2UgZXNj
YWxhdGlvbiBhdHRlbXB0czwvdGV4dD4KICA8dGV4dCB4PSI1MCIgeT0iMjQ2IiBmb250LWZhbWlse

T0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0iMTAiIGZpbGw9IiM5OTFiMWIiPlNwaWtlIG
luIDQwNHMg4oaSIERpcmVjdG9yeS9lbmRwb2ludCBlbnVtZXJhdGlvbjwvdGV4dD4KICA8dGV4dCB
4PSI1MCIgeT0iMjY0IiBmb250LWZhbWlseT0iQXJpYWwsIHNhbnMtc2VyaWYiIGZvbnQtc2l6ZT0i
MTAiIGZpbGw9IiM5OTFiMWIiPlNwaWtlIGluIDQyOXMg4oaSIERvUyBvciBhZ2dyZXNzaXZlIHNjc
mFwaW5nPC90ZXh0PgoKICA8IS0tIERRTCBRdWVyeSBTZWN0aW9uIC0tPgogIDxyZWN0IHg9IjQxMC
IgeT0iMTY1IiB3aWR0aD0iMzYwIiBoZWlnaHQ9IjEyMCIgcng9IjIzLz4
KICA8dGV4dCB4PSI1OTAiIHk9IjE4OCIgZm9udC1mYW1pbHk9IkFyaWFsLCBzYW5zLXNlcmlmIiBm
b250LXNpemU9IjExIiBmb250LXdlaWdodD0iYm9sZCIgZmlsbD0iIzk0YTNiOCIgdGV4dC1hbmNob
3I9Im1pZGRsZSI+UXVlcnkgU2VjdXJpdHkgU3RhdHVzIENvZGVzPC90ZXh0PgogIDx0ZXh0IHg9Ij
QyNSIgeT0iMjEwIiBmb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSI
jMjJjNTVlIj5mZXRjaDwvdGV4dD4KICA8dGV4dCB4PSI0NjMiIHk9IjIxMCIgZm9udC1mYW1pbHk9
Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0iI2Y4ZmFmYyI+c3BhbnM8L3RleHQ+CiAgP
HRleHQgeD0iNDI1IiB5PSIyMjgiIGZvbnQtZmFtaWx5PSJtb25vc3BhY2UiIGZvbnQtc2l6ZT0iMT
AiIGZpbGw9IiMxNDk2ZmYiPnwgZmlsdGVyPC90ZXh0PgogIDx0ZXh0IHg9IjQ4MCIgeT0iMjI4IiB
mb250LWZhbWlseT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjZjhmYWZjIj5pbiho
dHRwLnJlc3BvbnNlLnN0YXR1c19jb2RlLDwvdGV4dD4KICA8dGV4dCB4PSI0NDUiIHk9IjI0NiIgZ
m9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9udC1zaXplPSIxMCIgZmlsbD0iI2ZiYmYyNCI+ezQwMS
wgNDAzLCA0MjksIDUwMH0pPC90ZXh0PgogIDx0ZXh0IHg9IjQxMCIgeT0iMjQ2IiBmb250LWZhbWl
seT0ibW9ub3NwYWNlIiBmb250LXNpemU9IjEwIiBmaWxsPSIjMTQ5NmZmIj58IHN1bW1hcml6ZTwv
dGV4dD4KICA8dGV4dCB4PSI1MDUiIHk9IjI2NCIgZm9udC1mYW1pbHk9Im1vbm9zcGFjZSIgZm9ud
C1zaXplPSIxMCIgZmlsbD0iI2Y4ZmFmYyI+Y291bnQoKSwgYnk6e3N0YXR1c19jb2RlfTwvdGV4dD
4KPC9zdmc+)

```dql
// Analyze HTTP status code distribution
fetch spans
| filter isNotNull(http.response.status_code)
| summarize {count = count()}, by:{http.response.status_code}
| sort count desc
| limit 20
```

```dql
// Find security-relevant HTTP errors (401, 403, 429)
fetch spans
| filter in(http.response.status_code, {401, 403, 429})
| summarize {
    count = count()
  }, by:{http.response.status_code, dt.entity.service, http.route}
| sort count desc
| limit 50
```

```dql
// Security status codes over time (detect spikes)
fetch spans
```

```
| filter in(http.response.status_code, {401, 403, 429, 500})
| fieldsAdd time_bucket = bin(start_time, 10m)
| fieldsAdd status_category = if(http.response.status_code == 401,
"401_Unauthorized",
                               else: if(http.response.status_code == 403,
"403_Forbidden",
                               else: if(http.response.status_code == 429,
"429_RateLimited",
                               else: "500_ServerError")))
| summarize {count = count()}, by:{time_bucket, status_category}
| sort time_bucket asc, status_category
| limit 200
```

---

4. Authentication Failure Detection

Monitor authentication endpoints for potential brute force or credential stuffing attacks.

```dql
// Find 401 Unauthorized responses (failed authentication)
fetch spans
| filter http.response.status_code == 401
| fields start_time,
         dt.entity.service,
         http.request.method,
         http.route,
         url.path,
         trace.id
| sort start_time desc
| limit 100
```

```dql
// Count authentication failures by endpoint
fetch spans
| filter http.response.status_code == 401
| summarize {
    failure_count = count(),
    unique_traces = countDistinct(trace.id)
  }, by:{dt.entity.service, http.route}
| sort failure_count desc
| limit 30
```

```dql
```

```
// Authentication failure rate over time
fetch spans
| filter isNotNull(http.response.status_code)
| filter contains(http.route, "auth") or contains(http.route, "login") or
contains(span.name, "login")
| fieldsAdd time_bucket = bin(start_time, 10m)
| summarize {
    total_attempts = count(),
    failures = countIf(http.response.status_code == 401)
  }, by:{time_bucket}
| fieldsAdd failure_rate_pct = (failures * 100.0) / total_attempts
| sort time_bucket asc
| limit 50
```

```dql
// High-frequency authentication failures by service
fetch spans
| filter contains(span.name, "auth") or contains(span.name, "login")
| filter span.status_code == "error" or http.response.status_code == 401 or
http.response.status_code == 403
| summarize {
    failure_count = count()
  }, by:{dt.entity.service, span.name, http.response.status_code}
| filter failure_count > 10
| sort failure_count desc
```

---

5. Anomalous Traffic Patterns

Detect unusual patterns that might indicate attacks or misuse.

```dql
// Find endpoints with unusually high error rates
fetch spans
| filter span.kind == "server"
| filter isNotNull(http.response.status_code)
| summarize {
    total_requests = count(),
    error_4xx = countIf(http.response.status_code >= 400 and
http.response.status_code < 500),
    error_5xx = countIf(http.response.status_code >= 500)
  }, by:{dt.entity.service, http.route}
| fieldsAdd error_rate_4xx = (error_4xx * 100.0) / total_requests
| fieldsAdd error_rate_5xx = (error_5xx * 100.0) / total_requests
| filter error_rate_4xx > 20 or error_rate_5xx > 5
```

```
| sort error_rate_4xx desc
| limit 30
```

```dql
// Detect potential enumeration attacks (high 404 rates)
fetch spans
| filter http.response.status_code == 404
| summarize {
    not_found_count = count(),
    unique_paths = countDistinct(url.path)
  }, by:{dt.entity.service}
| filter not_found_count > 100
| sort not_found_count desc
| limit 20
```

```dql
// Find rate limiting events (429 responses)
fetch spans
| filter http.response.status_code == 429
| fields start_time,
         dt.entity.service,
         http.route,
         trace.id
| sort start_time desc
| limit 50
```

```dql
// Unusual access patterns - services with high error rates
fetch spans
| filter span.kind == "server"
| summarize {
    requests = count(),
    unique_operations = countDistinct(span.name),
    error_rate = (countIf(span.status_code == "error") * 100.0) / count()
  }, by:{dt.entity.service}
| filter error_rate > 20
| sort error_rate desc
```

---

6. Sensitive Endpoint Monitoring

Monitor access patterns to sensitive endpoints like admin panels,
configuration APIs, and user data endpoints.

```dql
// Find access to admin or configuration endpoints
fetch spans
| filter span.kind == "server"
| filter contains(url.path, "admin")
      or contains(url.path, "config")
      or contains(url.path, "settings")
      or contains(span.name, "admin")
| fields start_time,
         dt.entity.service,
         http.request.method,
         url.path,
         http.response.status_code,
         trace.id
| sort start_time desc
| limit 100
```

```dql
// Monitor data export or bulk operations
fetch spans
| filter span.kind == "server"
| filter contains(url.path, "export")
      or contains(url.path, "download")
      or contains(url.path, "bulk")
| summarize {
    access_count = count(),
    unique_traces = countDistinct(trace.id)
  }, by:{dt.entity.service, http.route}
| sort access_count desc
| limit 20
```

```dql
// Identify services that might handle regulated data
fetch spans
| filter contains(span.name, "health") or
        contains(span.name, "patient") or
        contains(span.name, "payment") or
        contains(span.name, "card") or
        contains(span.name, "billing")
| summarize {
    span_count = count(),
    unique_operations = countDistinct(span.name)
  }, by:{dt.entity.service}
| sort span_count desc
```

````
---

## 7. OpenPipeline for Data Masking

Use OpenPipeline to mask sensitive data **before** it is stored in Grail.

### Email Masking Configuration

```yaml
# Mask email addresses in URLs and attributes
processing:
  - type: replace
    field: http.url
    pattern: "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}"
    replacement: "***@***.***"
```

### Credit Card Masking

```yaml
# Mask credit card numbers (13-16 digits)
processing:
  - type: replace
    field: http.url
    pattern: "\\b[0-9]{13,16}\\b"
    replacement: "****-****-****-****"
```

### Token/API Key Masking

```yaml
# Mask tokens, keys, secrets in query parameters
processing:
  - type: replace
    field: http.url
    pattern: "(token|key|secret|password)=([^&]+)"
    replacement: "$1=***REDACTED***"
```

### Routing Sensitive Data to Restricted Buckets

```yaml
# Route sensitive spans to restricted buckets
routing:
  - condition: contains(service.name, "payment") or contains(service.name,
"auth")
    bucket: spans_sensitive
````
````

````
  - condition: matchesValue(deployment.environment, "production")
    bucket: spans_production

  - condition: true
    bucket: spans_default
```

### Verify Masking Is Working

```dql
// Verify email masking is working
// Result should be empty if masking is effective
fetch spans
| filter isNotNull(url.path)
| filter contains(url.path, "@") and not(contains(url.path, "***@"))
| fields url.path
| limit 10
```

---

## 8. Compliance Considerations

### GDPR Compliance

- **Data minimization**: Only collect necessary data
- **Purpose limitation**: Use data only for stated purposes
- **Storage limitation**: Define retention periods
- **Right to erasure**: Plan for data deletion requests

### PCI DSS Compliance

- **Never store** full credit card numbers
- **Mask** cardholder data in all traces
- **Encrypt** sensitive data at rest
- **Audit** access to payment-related traces

### HIPAA Compliance

- **Protect** PHI (Protected Health Information)
- **Encrypt** health-related span data
- **Restrict** access based on role
- **Audit** all access to healthcare traces

### Secure Query Patterns

**Use aggregations instead of exposing individual records:**
````
````

```dql
// SECURE: Aggregate patterns without exposing individual data
fetch spans
| filter span.kind == "server"
| summarize {
    request_count = count(),
    unique_routes = countDistinct(http.route),
    error_count = countIf(span.status_code == "error")
  }, by:{dt.entity.service}
| sort request_count desc
```

Use `http.route` (pattern) instead of `url.path` (full URL):

```dql
// SECURE: Use http.route (pattern) instead of url.path (full URL)
// http.route: /users/:id (pattern, no PII)
// url.path: /users/john.doe@email.com (actual value, potential PII)

fetch spans
| filter isNotNull(http.route)
| summarize {
    requests = count(),
    avg_ms = avg(duration) / 1000000
  }, by:{dt.entity.service, http.route, http.request.method}
| sort requests desc
| limit 20
```

Limit exposure of trace.id - get just enough to diagnose:

```dql
// For investigation: Get just enough trace.ids to diagnose
fetch spans
| filter span.status_code == "error"
| summarize {
    error_count = count(),
    sample_trace = takeFirst(trace.id)
  }, by:{dt.entity.service, span.name}
| sort error_count desc
| limit 10
```

---

9. Security Audit Queries

Use these queries for regular security audits.

```dql
// Security summary dashboard: Overall security status
fetch spans
| filter isNotNull(http.response.status_code)
| summarize {
    total_requests = count(),
    auth_failures_401 = countIf(http.response.status_code == 401),
    forbidden_403 = countIf(http.response.status_code == 403),
    rate_limited_429 = countIf(http.response.status_code == 429),
    server_errors_5xx = countIf(http.response.status_code >= 500)
  }
| fieldsAdd auth_failure_rate = (auth_failures_401 * 100.0) / total_requests
| fieldsAdd forbidden_rate = (forbidden_403 * 100.0) / total_requests
```

```dql
// Security events timeline
fetch spans
| filter in(http.response.status_code, {401, 403, 429})
| makeTimeseries {
    auth_failures = countIf(http.response.status_code == 401),
    forbidden = countIf(http.response.status_code == 403),
    rate_limited = countIf(http.response.status_code == 429)
  }, interval: 10m
```

```dql
// Services with highest security event rates
fetch spans
| filter isNotNull(http.response.status_code)
| summarize {
    total_requests = count(),
    security_events = countIf(in(http.response.status_code, {401, 403, 429}))
  }, by:{dt.entity.service}
| fieldsAdd security_event_rate = (security_events * 100.0) / total_requests
| filter security_events > 0
| sort security_event_rate desc
| limit 20
```

```dql
// Final audit: Summary of potential security concerns
fetch spans
| summarize {
    total_spans = count(),
    spans_with_url = countIf(isNotNull(url.path)),
```

```
    spans_with_db_query = countIf(isNotNull(db.statement)),
    spans_with_errors = countIf(span.status_code == "error"),
    auth_failures = countIf(
        (contains(span.name, "auth") or contains(span.name, "login")) and
        (span.status_code == "error" or http.response.status_code >= 400))
  }
| fieldsAdd url_percent = (spans_with_url * 100.0) / total_spans
| fieldsAdd db_percent = (spans_with_db_query * 100.0) / total_spans
```

---

10. Security Checklist

Data Protection

- [ ] Audit spans for PII in URLs, headers, query params
- [ ] Configure OpenPipeline to mask sensitive data
- [ ] Use `http.route` instead of `url.path` when possible
- [ ] Drop or mask database query contents
- [ ] Sanitize error messages and stack traces

Access Control

- [ ] Route sensitive spans to restricted buckets
- [ ] Configure IAM policies for span access
- [ ] Limit who can query raw span data
- [ ] Use aggregations instead of exposing individual records

Compliance

- [ ] Define data retention policies per bucket
- [ ] Document data processing for GDPR Article 30
- [ ] Ensure PCI DSS compliance for payment spans
- [ ] Protect PHI for HIPAA compliance

Monitoring

- [ ] Regular audits for new PII exposure
- [ ] Monitor authentication failure patterns
- [ ] Alert on unusual access patterns
- [ ] Review masking effectiveness periodically

Summary

In this notebook, you learned:

✅ **Understanding sensitive data locations** in span attributes
✅ **PII audit queries** to find emails, tokens, and credentials in spans
✅ **HTTP status code analysis** for security-relevant codes (401, 403, 429, 500)
✅ **Authentication failure detection** to identify potential attacks
✅ **Anomalous traffic pattern detection** for enumeration and abuse
✅ **Sensitive endpoint monitoring** for admin and data access
✅ **OpenPipeline masking** configurations for emails, credit cards, tokens
✅ **Compliance considerations** for GDPR, PCI DSS, and HIPAA
✅ **Security audit queries** for compliance and reporting
✅ **Security checklist** for ongoing protection

Next Steps

Continue to **SPANS-07: Grail Buckets & OpenPipeline** to learn:
- Understanding Grail bucket architecture
- Configuring OpenPipeline for span processing
- Data routing and retention strategies
- Access control for span data