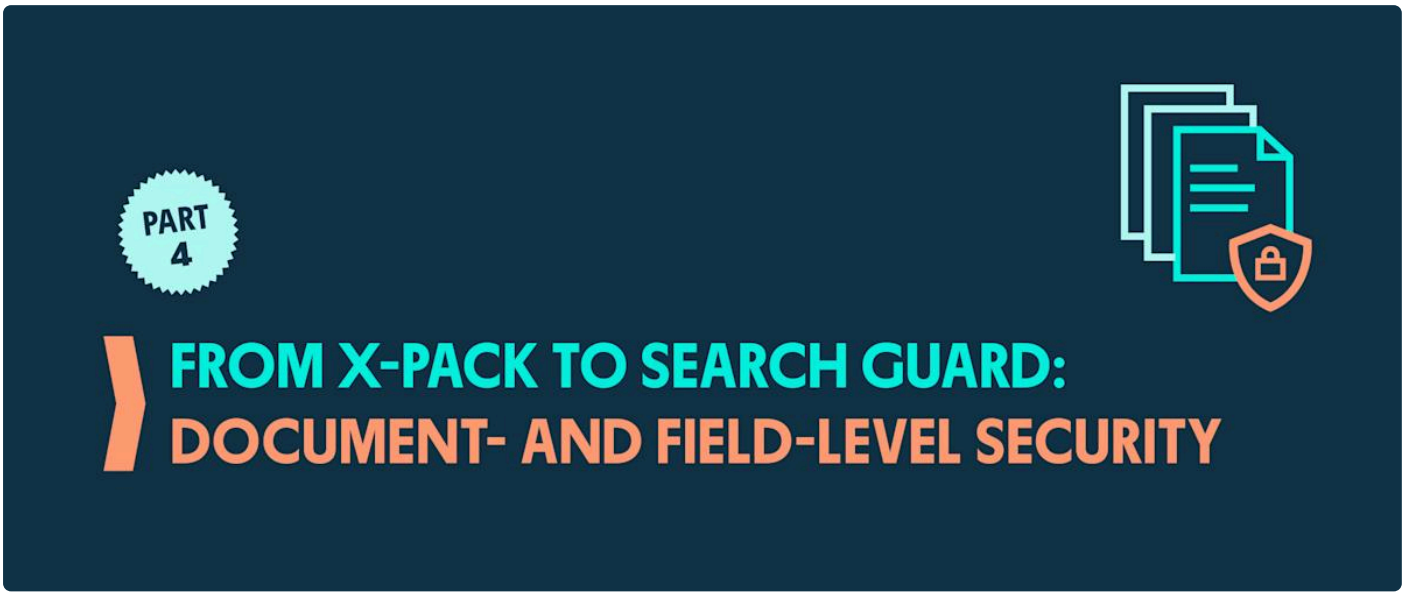
From X-Pack to Search Guard: Part 4 - Document- and Field-Level Security

Jochen Kressin · 2025-07-16



In one of our previous articles, we discussed how to migrate simple roles from X-Pack Security to Search Guard. Now, we'll focus on migrating Document Level Security (DLS) and Field Level Security (FLS) configurations, which provide fine-grained access control within your Elasticsearch deployment.

X-Pack DLS and FLS

In X-Pack Security, Document Level Security and Field Level Security are powerful features that allow you to restrict which documents and fields users can access based on their roles.

Here's an example of an X-Pack role with both DLS and FLS applied:

```
{
  "cluster":[
    "all"
  ],
  "indices":[
    {
      "names":[
        "index*",
        "some"
      ],
      "privileges":[
        "all"
      ],
      "query": {
        "match": {
          "category": "mycategory"
        }
      },
      "field_security": {
        "grant": [ "name", "phone*", "category", "product.asin" ]
      }
    }
  ]
}
```

The `query` element restricts access to only those documents that match the specified query, effectively implementing DLS. Meanwhile, the `field_security` element limits which fields users can see, implementing FLS by granting access only to fields that match the listed patterns.

Search Guard DLS and FLS

Search Guard provides excellent Document Level Security and Field Level Security features that work similarly to X-Pack's implementation, but with some syntax differences.

A Search Guard role with equivalent DLS and FLS settings would look like this:

```
{
  "cluster_permissions":[
    "SGS_CLUSTER_ALL"
  ],
  "index_permissions":[
    {
      "index_patterns":[
        "index*",
        "some"
      ],
      "allowed_actions":[
        "SGS_INDICES_ALL"
      ],
      "_dls_": "{\"match\": { \"category\": \"mycategory\"}}",
      "_fls_": [ "name", "phone*", "category", "product.asin" ]
    }
  ]
}
```

Migrating DLS and FLS

To migrate your X-Pack DLS and FLS settings to Search Guard, you'll need to adapt the syntax according to this mapping:

X-Pack Security	Search Guard
`query`	`_dls_`
`field_security`	`_fls_`

The content of the `query` and `field_security` fields can generally be used without significant modifications. Search Guard also supports wildcard syntax and dot notation for accessing nested fields.

Templating and Attribute-based Security

Both X-Pack and Search Guard support parameterized DLS queries for dynamic security rules based on user attributes. X-Pack calls this feature "Templating," while Search Guard refers to it as "Dynamic DLS Queries."

User Attributes in Search Guard

In Search Guard, user attributes can come from various sources such as LDAP, JWT claims, or the internal user database. To make these attributes available for DLS queries, you must properly configure them in `sg_authc.yml`.

For the internal user database, attributes are defined in `sg_internal_users.yml`:

```
johndoe:
  hash: "$2a$12$..."
  attributes:
    department: "finance"
    category: "vip"
    access_level: 10
```

However, these attributes aren't automatically available for DLS queries. You need to map them in `sg_authc.yml` using the `user_mapping.attributes` configuration:

```
auth_domains:
- type: basic/internal_users_db
  user_mapping.attributes.from:
    department: user_entry.attributes.department
    category: user_entry.attributes.category
    access_level: user_entry.attributes.access_level
```

This mapping makes the attributes available to DLS queries as `${user.attrs.department}`, `${user.attrs.category}`, etc.

Migrating X-Pack Templating to Search Guard

In X-Pack, a templated DLS query might look like:

```
"query":{
  "template":{
    "source":{
      "match":{
        "category":"{{_user.metadata.category}}"
      }
    }
  }
}
```

The equivalent in Search Guard would be:

```
"_dls_": "{\"match\": {\"category\": \"${user.attrs.category}\"}}"
```

Notice the following key differences:

- X-Pack uses Mustache syntax with double curly braces `{{...}}`
- Search Guard uses variable substitution with `${...}` notation
- User attributes in Search Guard are accessed via `user.attrs.attribute_name`

Use the following mapping table to adjust X-Pack metadata variables to Search Guard variables:

X-Pack Security	Search Guard
`_user.username`	`${user.name}`
`_user.roles`	`${user.roles}`
`_user.metadata.XXX`	`${user.attrs.XXX}`

Example: Complete Migration Workflow

Original X-Pack user with metadata:

```
  {
    "username": "johndoe",
    "roles": ["analyst"],
    "metadata": {
      "department": "finance",
      "category": "vip"
    }
  }
```

Migrated user in Search Guard's internal user database (`sg_internal_users.yml`):

```
  johndoe:
    hash: "$2a$12$..."
    backend_roles: ["analyst"]
    attributes:
      department: "finance"
      category: "vip"
```

Attribute mapping in `sg_authc.yml`:

```
  auth_domains:
  - type: basic/internal_users_db
    user_mapping.attributes.from:
      department: user_entry.attributes.department
      category: user_entry.attributes.category
```

X-Pack role with templated DLS query:

```
  "query":{
    "template":{
      "source":{
        "bool": {
          "must": [
            {"match": {"department": "{{_user.metadata.department}}"}},
            {"match": {"customer_type": "{{_user.metadata.category}}"}}
          ]
        }
      }
    }
  }
```

Migrated Search Guard role with dynamic DLS query:

```
  "_dls_": "{\"bool\": {\"must\": [{\"match\": {\"department\": \"${user.attrs.department}\"}},
{\"match\": {\"customer_type\": \"${user.attrs.category}\"}}]}}"
```

This approach ensures that user attributes from the internal database are properly mapped and available for DLS queries in Search Guard.

Advanced Features and Limitations

Search Guard offers some advantages over X-Pack when it comes to DLS and FLS:

1. **Mixed Include/Exclude Fields**: Unlike X-Pack, Search Guard allows mixing included and excluded fields in FLS with a simple syntax. To exclude fields in Search Guard, prefix the field name with a tilde (`~`):

```
"_fls_": [ "name", "address", "~creditcard", "~ssn" ]
```

2. **Field Masking**: Search Guard also supports field masking (anonymization), allowing you to display sensitive fields in a hashed format rather than removing them completely:

```
"_masked_fields_": [ "phone", "email" ]
```

Articles in this series

- From X-Pack to Search Guard: Part 1 - TLS Setup
- From X-Pack to Search Guard: Part 2 - Roles and Permissions
- From X-Pack to Search Guard: Part 3 - Users
- From X-Pack to Search Guard: Part 4 - Document- and Field-Level Security

---

This article was published on the Search Guard blog: https://search-guard.com/blog/from-x-pack-to-search-guard-part-4-document-and-field-level-security/