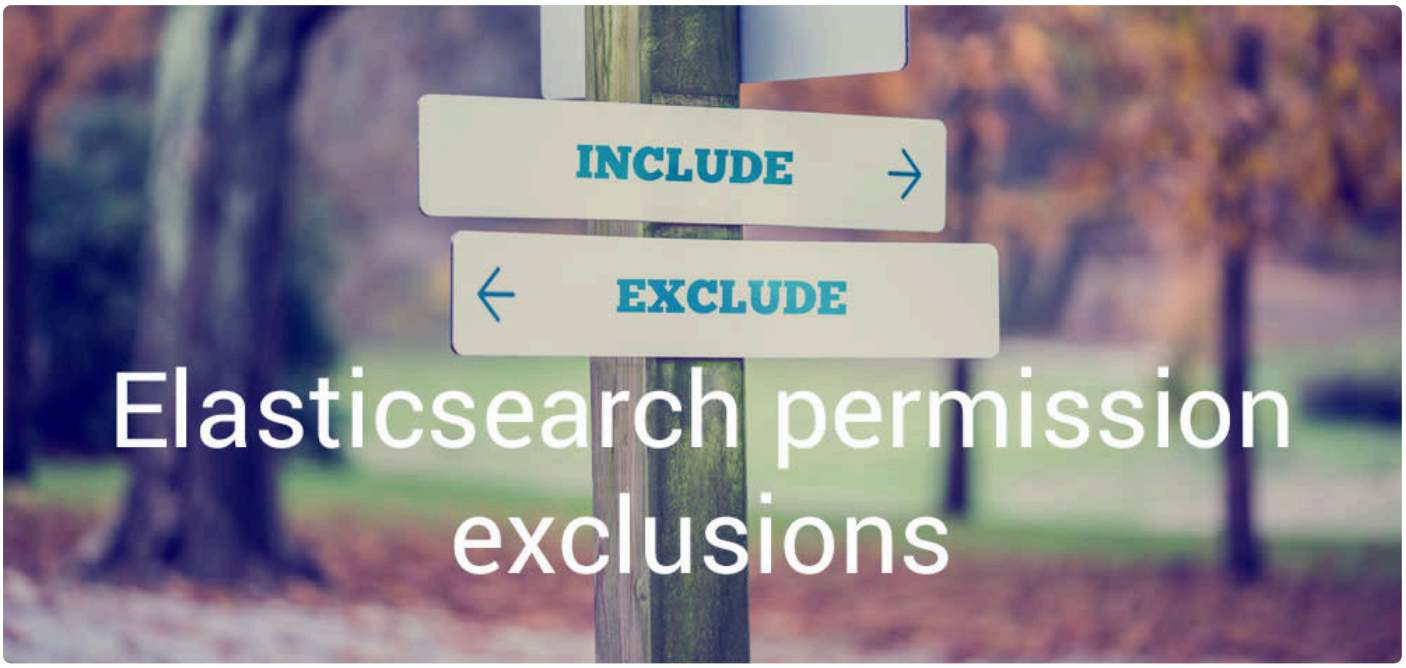
Exclude Elasticsearch permissions from Search Guard roles

Jochen Kressin · 2020-11-18



When defining access controls for Elasticsearch, you can use wildcards and regular expressions in index names for any Search Guard role. This allows for flexible and dynamic role definitions: You can grant a user access to indices that follow a specific naming pattern by defining a single role.

This is handy, but sometimes also too broad. You may want to grant a user access to a wide range of indices, but at the same time exclude specific ones. Or, grant read/write permissions to indices, but them limit the permissions to read-only for specific indices.

Search Guard v46 and above allows combining including and excluding indices and permissions in a single role definition.

Using wildcards to grant index access

Assume we have indices which follow this naming pattern like:

```
logs-<systemname>-<date>-<production|staging|development>
```

For example:

```
logs-webshop-20201115-production
logs-webshop-20201115-staging
logs-transactions-20201115-production
logs-transactions-20201115-staging
```

You can define a role that has access to all *logs* indices easily by using a wildcard like:

```
my_sg_role:
  cluster_permissions:
    - *
  index_permissions:
    - index_patterns:
        - "logs-*"
      allowed_actions:
        - SGS_CRUD
```

Excluding specific indices

Let's say you want to keep the role definition as broad as above but explicitly deny access to any *transactions* index from *production*. This would mean to either using multipe roles, or using complex regular expressions.

Instead, you now can simply extend the role definition with an *excludeindexpermissions* section:

```
my_sg_role:
  cluster_permissions:
    - *
  index_permissions:
    - index_patterns:
        - "logs-*"
      allowed_actions:
        - SGS_CRUD
  exclude_index_permissions:
    - index_patterns:
       - "logs-transactions-*-production"
      actions:
       - "*"
```

This role definition allows *CRUD* operations on any index following the pattern *logs-**. In addition, it removes all permissions for any index following the pattern *logs-transactions-*-production*.

"Grant access to everything, but ..."

Another use case is to grant users access to all data in your Elasticsearch cluster but exclude specific indices that may contain sensitive data.

The role definition for this is also straight-forward:

```
all_access_but:
  cluster_permissions:
    - *
  index_permissions:
    - index_patterns:
        - "*"
      allowed_actions:
        - SGS_CRUD
 exclude_index_permissions:
    - index_patterns:
       - "secret-index-*"
      actions:
       - "*"
```

This role first grants CRUD access to all indices in the cluster. The *excludeindexpermissions* section then removes all permissions and thus all access from the indices starting with *secret-index-**

Selectively removing access permissions

In the examples above, we have used a wildcard in the *actions* list of the *excludeindexpermissions* section. This removes all permissions from the index pattern, which means the role cannot access the index at all.

We can also remove permissions selectively on a fine-grained level. Continuing with the example above, we want to grant the user CRUD access to all indices but disallow deleting data from any *production* index. A role definition would look like this:

```
my_sg_role:
  cluster_permissions:
    - *
  index_permissions:
    - index_patterns:
        - "logs-*"
      allowed_actions:
        - SGS_CRUD
  exclude_index_permissions:
    - index_patterns:
       - "logs-*-production"
      actions:
       - DELETE
```

With this role definition, the user has CRUD permissions to any index that starts with *logs-**, but we remove the DELETE permission if the index name has the pattern *logs-*-production*.

Combining multiple roles

Permission exclusions do not only affect permissions granted in the same role. Instead, permission exclusions also affect permissions granted by other roles. Thus, you can grant permissions in one role and use additional roles to selectively remove some of the permissions.

For example, you can first create a very broad "all-access" role and use the role mapping to assign this role to all users:

```
all_access:
  cluster_permissions:
    - *
  index_permissions:
    - index_patterns:
        - "*"
      allowed_actions:
        -  "*"
```

Then you can define roles that exclude certain index patterns and actions, and map these roles to your users in addition:

```
exclude_production:
  exclude_index_permissions:
    - index_patterns:
       - "logs-*-production"
      actions:
       - "*"
```

```
exclude_staging:
  exclude_index_permissions:
    - index_patterns:
       - "logs-*-staging"
      actions:
       - "*"
```

Summary

In addition to allowing users access to indices and data in Elasticsearch, you can also explicitly exclude indices and actions.

By combining these two features, you can create very flexible role definitions and avoid repetition.

Where to go next

- Read the documentation on permission exclusion
- If you have any question, hop over to our forum and ask us

Image: Shutterstock / Gajus

This article was published on the Search Guard blog: https://search-guard.com/blog/elasticsearch-exclude-permissions/