User Impersonation: Submit requests on behalf of another user

Cliff Staley · 2018-08-08

The best way to assure that security rules work as desired, is to test the configuration. Fortunately Search Guard comes with User Impersonation feature that makes testing easy.

When do we need to impersonate the user? Assume that Jack has been granted permissions on an Elasticsearch cluster. If Jack encounters some issues with accessing the cluster, and admin user can login to the cluster on behalf of Jack. This is what user impersonation is and it allows admin to check if Jack's permissions work properly and debug problems.

The same feature can be used for penetration testing. If Jack has been granted only read permissions, we can check if he is able to delete or alter data.

Let us see some example on a demo Search Guard installation with the Shakespeare dataset loaded. "To be, or not be" *phrases* can be searched by the *readall* user that comes with Search Guard demo installation:

```
curl -u readall:readall -k 'https://localhost:9200/shakespeare/_search?
q=To+be+or+not+to+be&size=1'
```

Which gives us:

```
{
   "took":16,
   "timed_out":false,
   "_shards":{
      "total":5,
      "successful":5,
      "skipped":0,
      "failed":0
   },
   "hits":{
      "total":30384,
      "max_score":17.413345,
      "hits":[
         {
            "_index":"shakespeare",
            "_type":"doc",
            "_id":"34229",
            "_score":17.413345,
            "_source":{
               "type":"line",
               "line_id":34230,
               "play_name":"Hamlet",
               "speech_number":19,
               "line_number":"3.1.64",
               "speaker":"HAMLET",
               "text_entry":"To be, or not to be: that is the question:"
            }
         }
      ]
   }
}
```


In this case, we knew password of the *readall* user which is also *readall*. User impersonation allows us to achieve the same results without knowing the password of *readall* user.

To make it work, we add following lines to *elasticsearch.yml* and restart the cluster:

```
searchguard.authcz.rest_impersonation_user.admin:
  - "readall"
```

This tells Search Guard that the user *admin* is allowed to impersonate as user *readall*. You can specify multiple users here, and wildcards are also supported. To allow the *admin* user to impersonate as any other user, we can simply use:

```
searchguard.authcz.rest_impersonation_user.admin:
  - "*"
```

With this configuration we can now try to use the *admin* user to impersonate as *readall*:

```
curl -u admin:admin -H "sg_impersonate_as: readall" -k
'https://localhost:9200/shakespeare/_search?q=To+be+or+not+to+be&size=1'
```

We used an HTTP header *sg_impersonate_as* to specify which user we want to impersonate as. The request is now performed as *readall*. As expected, we get the same result from Elasticsearch as before.

But wait - the admin role has full access to all indices in the cluster. How do we know that impersonation really worked? Let us check if we are able to delete the index:

```
curl -u admin:admin -H "sg_impersonate_as: readall" -k -XDELETE
'https://localhost:9200/shakespeare/'
```

As expected a security exception is raised since the *readall* user has no permission to delete any index:

```
{
   "error":{
      "root_cause":[
         {
            "type":"security_exception",
            "reason":"no permissions for [indices:admin/delete] and User [name=readall, roles=
[readall], requestedTenant=null]"
         }
      ],
      "type":"security_exception",
      "reason":"no permissions for [indices:admin/delete] and User [name=readall, roles=[readall],
requestedTenant=null]"
   },
   "status":403
}
```

We can also check the Search Guard audit log to see that operation. The generated *MISSING_PRIVILEGES* event contains two user entries. The *audit_request_initiating_user* is admin, since we used admin to initiate the request. The *audit_request_effective_user* is *readall*, since the request was performed in the context of the user *readall*.

t	_id	Y2lOCWUBQ8Fy2hl5vyPb
t	_index	sg6-auditlog-2018.08.05
#	_score	-
t	_type	auditlog
t	audit_category	MISSING_PRIVILEGES
#	audit_format_version	3
t	audit_node_host_address	127.0.0.1
t	audit_node_host_name	127.0.0.1
t	audit_node_id	uuBn12_KRGqPUnI_xzcZjg
t	audit_node_name	uuBn12_
t	audit_request_effective_user	readall
t	audit_request_layer	TRANSPORT
t	audit_request_origin	REST
?	audit_request_privilege	⚠ indices:admin/delete
t	audit_request_remote_address	127.0.0.1
?	audit_trace_indices	⚠ shakespeare
?	audit_trace_resolved_indices	⚠ shakespeare
?	audit_trace_task_id	⚠ uuBn12_KRGqPUnI_xzcZjg:991
?	audit_transport_request_type	⚠ DeleteIndexRequest
⏲	audit_utc_timestamp	August 5th 2018, 10:58:00.533

Summary

With Search Guard User Impersonation it is possible to submit requests on behalf of another user. This makes it possible to debug permissions problems and for penetration testing.

Where to go next?

- Download and install the latest Search Guard version
- Read our official documentation about User Impersonation
- If you have questions, ask on Google Groups
- Follow us on twitter for getting the latest Search Guard news

Image: shutterstock / Minerva Studio

This article was published on the Search Guard blog: https://search-guard.com/blog/elasticsearch-user-impersonation/
Search Guard is Security and Alerting for Elasticsearch and Kibana — search-guard.com · Start a free trial