ONTOTEXT ANSWER:

As you are most likely well aware, GraphDB has support for RBAC. You can enable read and write permissions on the repository level. Besides this, you can also fine-tune permissions on different REST APIs.

However, before we go over “how”, let’s have quick look at “why”. GraphDB has security over all its endpoints by default. Our security is exclusive – this means that each endpoint that is not explicitly whitelisted will be inaccessible. There is usually no need to alter the security. A few cases when you might consider this is:

- When there is a security vulnerability on a given endpoint and you want to forbid it outright. If that is the case, you should also let us know about the vulnerability. We are devoted to securing GraphDB for any fresh issues and would happily release a patch that covers the vulnerabilities without impacting the database’s functionality.
- You want to alter how GraphDB operates. The only reason for doing this is that you want some very specific functionality such as, for example, your repository manager being able to edit the cluster. That’s an unusual course of action.
- You have extended the database (e.g., with a plugin) in a way that requires a new endpoint – at the moment, the open-source plugin API isn’t that flexible. However, if it is, at some point in the future, you may want to set up special permissions for your new plugin.

Now, on to “how” to edit our security module. GraphDB implements Spring Security. This means that hidden deep within our library files, there is a configuration XML that you can edit. By default, the file is located at `lib/common/WEB-INF/classes/META-INF/spring`. It contains all GraphDB security configurations.

Here’s an example:

```
<!-- Saved graphs retrieve all users -->
<security:intercept-url pattern="/rest/explore-graph/saved/**" method="GET" access="IS_AUTHENTICATED_FULLY" />
<!-- Saved graphs modification only users with write access -->
<security:intercept-url pattern="/rest/explore-graph/saved/**" access="ROLE_USER" />
```

This means that any authenticated user can fetch (GET) a saved visual graph. However, only users with full access can modify their own saved graphs. Maybe you want to edit this to hide

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

the saved graphs. That is as easy as dropping the first rule. With Spring security, the most permissive filter takes precedence. So, a user who is both authenticated fully and has the ROLE_USER can alter their own saved graphs.

One useful tool which you can use when you want to disable an endpoint is the denyAll() method. This can be used when you want to remove some GraphDB capabilities.

```
<!-- Everything within /rest not explicitly allowed above should be denied. -->
<security:intercept-url pattern="/rest/**" access="denyAll()" />
```

One practical example. If someone wants to use anonymous access for GraphDB, but still prevent users from obtaining a database dump by using the “Export” button under “Explore” -> “Graphs Overview”, they can add the following line to their configuration.

```
<security:intercept-url pattern="/repositories/*/statements" access="ROLE_USER" />
```

Of course, since this file is located within our library directory, when you upgrade your GraphDB, you would have to reconfigure your security settings.

Help us get better: review GraphDB on Gartner

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com