Red Hat build of Quarkus 3.8

Security overview

Last Updated: 2024-07-24

Legal Notice

Abstract

Quarkus Security provides a complete framework for Java application security, incorporating diverse authentication methods, role-based access control (RBAC), and features like SSL/TLS. This guide covers its main functionalities, setup, and testing, targeting developers integrating security into Quarkus applications.

Table of Contents

PROVIDING FEEDBACK ON RED HAT BUILD OF QUARKUS DOCUMENTATION

To report an error or to improve our documentation, log in to your Red Hat Jira account and submit an issue. If you do not have a Red Hat Jira account, then you will be prompted to create an account.

Procedure

1. Click the following link to **create a ticket**.
2. Enter a brief description of the issue in the **Summary**.
3. Provide a detailed description of the issue or enhancement in the **Description**. Include a URL to where the issue occurs in the documentation.
4. Clicking **Submit** creates and routes the issue to the appropriate documentation team.

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. QUARKUS SECURITY OVERVIEW

Quarkus Security is a framework that provides the architecture, multiple authentication and authorization mechanisms, and other tools to build secure and production-quality Java applications.

Before building security into your Quarkus applications, learn about the Quarkus Security architecture and the different authentication mechanisms and features you can use.

1.1. KEY FEATURES OF QUARKUS SECURITY

The Quarkus Security framework provides built-in security authentication mechanisms for Basic, Form-based, and mutual TLS (mTLS) authentication. You can also use other well-known authentication mechanisms, such as OpenID Connect (OIDC). Authentication mechanisms depend on Identity providers to verify the authentication credentials and map them to a **SecurityIdentity** instance with the username, roles, original authentication credentials, and other attributes.

Red Hat build of Quarkus also includes built-in security to allow for role-based access control (RBAC) based on the common security annotations **@RolesAllowed**, **@DenyAll**, **@PermitAll** on REST endpoints, and Contexts and Dependency Injection (CDI) beans. For more information, see the Quarkus Authorization of web endpoints guide.

Quarkus Security also supports the following features:

- Proactive authentication
- Secure connections with SSL/TLS
- Cross-origin resource sharing
- Cross-Site Request Forgery (CSRF) prevention
- SameSite cookies
- Secrets engines
- Secure auto-generated resources by REST Data with Panache
- Secure serialization
- Security vulnerability detection and National Vulnerability Database (NVD) registration

Quarkus Security is also highly customizable. For more information, see the Quarkus Security tips and tricks guide.

1.2. GETTING STARTED WITH QUARKUS SECURITY

To get started with security in Quarkus, consider securing your Quarkus application endpoints with the built-in Quarkus Basic authentication and the Jakarta Persistence identity provider and enabling role-based access control.

Complete the steps in the Getting started with Security by using Basic authentication and Jakarta Persistence tutorial.

After successfully securing your Quarkus application with Basic authentication, you can increase the security further by adding more advanced authentication mechanisms, for example, the Quarkus OpenID Connect (OIDC) authorization code flow mechanism guide.

1.3. QUARKUS SECURITY TESTING

For guidance on testing Quarkus Security features and ensuring that your Quarkus applications are securely protected, see the Security testing guide.

1.4. MORE ABOUT SECURITY FEATURES IN QUARKUS

1.4.1. Cross-origin resource sharing

To make your Quarkus application accessible to another application running on a different domain, you need to configure cross-origin resource sharing (CORS). For more information about the CORS filter Quarkus provides, see the CORS filter section of the Quarkus "Cross-origin resource sharing" guide.

1.4.2. Cross-Site Request Forgery (CSRF) prevention

Quarkus Security provides a RESTEasy Reactive filter that can protect your applications against a Cross-Site Request Forgery attack. For more information, see the Quarkus Cross-Site Request Forgery Prevention guide.

1.4.3. SameSite cookies

You can add a SameSite cookie property to any of the cookies set by a Quarkus endpoint. For more information, see the SameSite cookies section of the Quarkus "HTTP reference" guide.

1.4.4. Secrets engines

You can use secrets engines with Quarkus to store, generate, or encrypt data.

Quarkus provides additional extensions in Quarkiverse for securely storing credentials, for example, Quarkus and HashiCorp Vault.

1.5. SECRETS IN ENVIRONMENT PROPERTIES

Quarkus provides support to store secrets in environment properties. For more information, see the Quarkus store secrets in an environment properties file guide.

1.5.1. Secure serialization

If your Quarkus Security architecture includes RESTEasy Reactive and Jackson, Quarkus can limit the fields included in JSON serialization based on the configured security. For more information, see the JSON serialization section of the Quarkus “Writing REST services with RESTEasy Reactive” guide.

1.5.2. Secure auto-generated resources by REST Data with Panache

If you use the REST Data with Panache extension to auto-generate your resources, you can still use security annotations within the package **jakarta.annotation.security**. For more information, see the Securing endpoints section of the Quarkus "Generating Jakarta REST resources with Panache" guide.

1.6. SECURITY VULNERABILITY DETECTION

Most Quarkus tags get reported in the US National Vulnerability Database (NVD). For information about security vulnerabilities, see the Security vulnerability detection and reporting in Quarkus guide.

1.7. REFERENCES

- Basic authentication
- Getting started with Security by using Basic authentication and Jakarta Persistence
- Protect a web application by using OIDC authorization code flow
- Protect a service application by using OIDC Bearer token authentication