Quarkus - Security Architecture and Guides

Quarkus Security provides the architecture, multiple authentication and authorization mechanisms, and other tools for the developers to build a production-quality security for their Quarkus applications.

This document provides a brief overview of Quarkus Security and links to the individual guides.

Architecture

`HttpAuthenticationMechanism` is the main entry into Quarkus HTTP Security.

Quarkus Security Manager uses `HttpAuthenticationMechanism` to extract the authentication credentials from the HTTP request and delegates to `IdentityProvider` to complete the conversion of these credentials to `SecurityIdentity`.

For example, the credentials may be coming with the HTTP `Authorization` header, client HTTPS certificates or cookies.

`IdentityProvider` verifies the authentication credentials and maps them to `SecurityIdentity` which contains the user name, roles, the original authentication credentials, and other attributes.

For every authenticated resource, you can inject a `SecurityIdentity` instance to get the authenticated identity information.

In some other contexts you may have other parallel representations of the same information (or parts of it) such as `SecurityContext` for JAX-RS or `JsonWebToken` for JWT.

Authentication mechanisms

Quarkus supports several sources to load authentication information from.

Basic and Form Authentication Mechanisms

Basic and Form HTTP-based authentication mechanisms are the core authentication mechanisms supported in Quarkus. Please see Basic HTTP Authentication and Form HTTP Authentication for more information.

Mutual TLS Authentication

Quarkus provides Mutual TLS authentication so that you can authenticate users based on their X.509 certificates.

Please see Mutual TLS Authentication for more information.

OpenId Connect

`quarkus-oidc` extension provides a reactive, interoperable, multi-tenant enabled OpenId Connect adapter which supports `Bearer Token` and `Authorization Code Flow` authentication mechanisms.

`Bearer Token` mechanism extracts the token from HTTP `Authorization` header. `Authorization Code Flow` mechanism uses OpenId Connect Authorization Code flow. It redirects the user to IDP to authenticate and completes the authentication process after the user has been redirected back to Quarkus by exchanging the provided code grant for ID, access and refresh tokens.

ID and access `JWT` tokens are verified with the refreshable `JWK` key set but both JWT and opaque (binary) tokens can be introspected remotely.

See the Using OpenID Connect to Protect Service Applications guide for more information about `Bearer Token` authentication mechanism.

See the Using OpenID Connect to Protect Web Application guide for more information about `Authorization Code Flow` authentication mechanism.

> ℹ️ Both `quarkus-oidc Bearer` and `Authorization Code Flow` Authentication mechanisms use SmallRye JWT to represent JWT tokens as Microprofile JWT `org.eclipse.microprofile.jwt.JsonWebToken`.

See Using OpenID Connect Multi-Tenancy for more information about multiple tenants which can support `Bearer` or `Authorization Code Flow` authentication mechanism and configured statically or dynamically.

If you use Keycloak and Bearer tokens then also see the Using Keycloak to Centralize Authorization guide.

SmallRye JWT

`quarkus-smallrye-jwt` provides Microprofile JWT 1.1.1 implementation and many more options to verify signed and encrypted `JWT` tokens and represent them as `org.eclipse.microprofile.jwt.JsonWebToken`.

It provides an alternative to `quarkus-oidc` Bearer Token Authentication Mechanism. It can currently verify only `JWT` tokens using the PEM keys or refreshable `JWK` key set.

Additionally it provides `JWT Generation API` for creating `signed`, `inner-signed` and/or `encrypted` JWT tokens with ease.

See the Using SmallRye JWT guide for more information.

OAuth2

`quarkus-elytron-security-oauth2` provides an alternative to `quarkus-oidc` Bearer Token

Authentication Mechanism. It is based on `Elytron` and is primarily meant for introspecting the opaque tokens remotely.

See the Using OAuth2 guide for more information.

LDAP

Please see the Authenticate with LDAP guide for more information about LDAP authentication mechanism.

Identity Providers

`IdentityProvider` converts the authentication credentials provided by `HttpAuthenticationMechanism` to `SecurityIdentity`.

Some extensions such as `OIDC`, `OAuth2`, `SmallRye JWT`, `LDAP` have the inlined `IdentityProvider` implementations which are specific to the supported authentication flow. For example, `quarkus-oidc` uses its own `IdentityProvider` to convert a token to `SecurityIdentity`.

If you use `Basic` or `Form` HTTP-based authentication then you have to add an `IdentityProvider` which can convert a user name and password to `SecurityIdentity`.

See JPA IdentityProvider and JDBC IdentityProvider for more information.

Combining Authentication Mechanisms

One can combine multiple authentication mechanisms if they get the authentication credentials from the different sources. For example, combining built-in `Basic` and `quarkus-oidc Bearer` authentication mechanisms is allowed, but combining `quarkus-oidc Bearer` and `smallrye-jwt` authentication mechanisms is not allowed because both will attempt to verify the token extracted from the HTTP `Authorization Bearer` scheme.

Proactive Authentication

By default, Quarkus does what we call proactive authentication. This means that if an incoming request has a credential then that request will always be authenticated (even if the target page does not require authentication).

See Proactive Authentication for more information.

Authorization

See Security Authorization for more information about Role Based Access Control and other authorization options.

Customization and other useful tips

Quarkus Security is highly customizable. One can register custom `HttpAuthenticationMechanism`s, `IdentityProvider`s and `SecurityidentityAugmentor`s.

See Security Customization for more information about customizing Quarkus Security and other useful tips about the reactive security, registering the security providers, etc.

Testing

See Security Testing for more information about testing Quarkus Security.

Secret Engines

Quarkus provides a very comprehensive HashiCorp Vault support, please see the Quarkus and HashiCorp Vault documentation for more information.