Financial-grade API (FAPI) support in Keycloak SSO

Marek Posolda

About me

- Working as software engineer in Red Hat since 2009
- Working on Keycloak project since 2013
- Graduate of FI MUNI

Keycloak - main features

- Authentication for web (and other) applications
- SSO - login once to all the applications
- Single-sign out
- Standards based (OpenID Connect, OAuth2, SAML2)

Identity management

- Realms, clients, roles, users, authorization data
- Storing data in RDBMS

UI

- Admin console
- Account management
- User login forms, registrations, reset passwords...

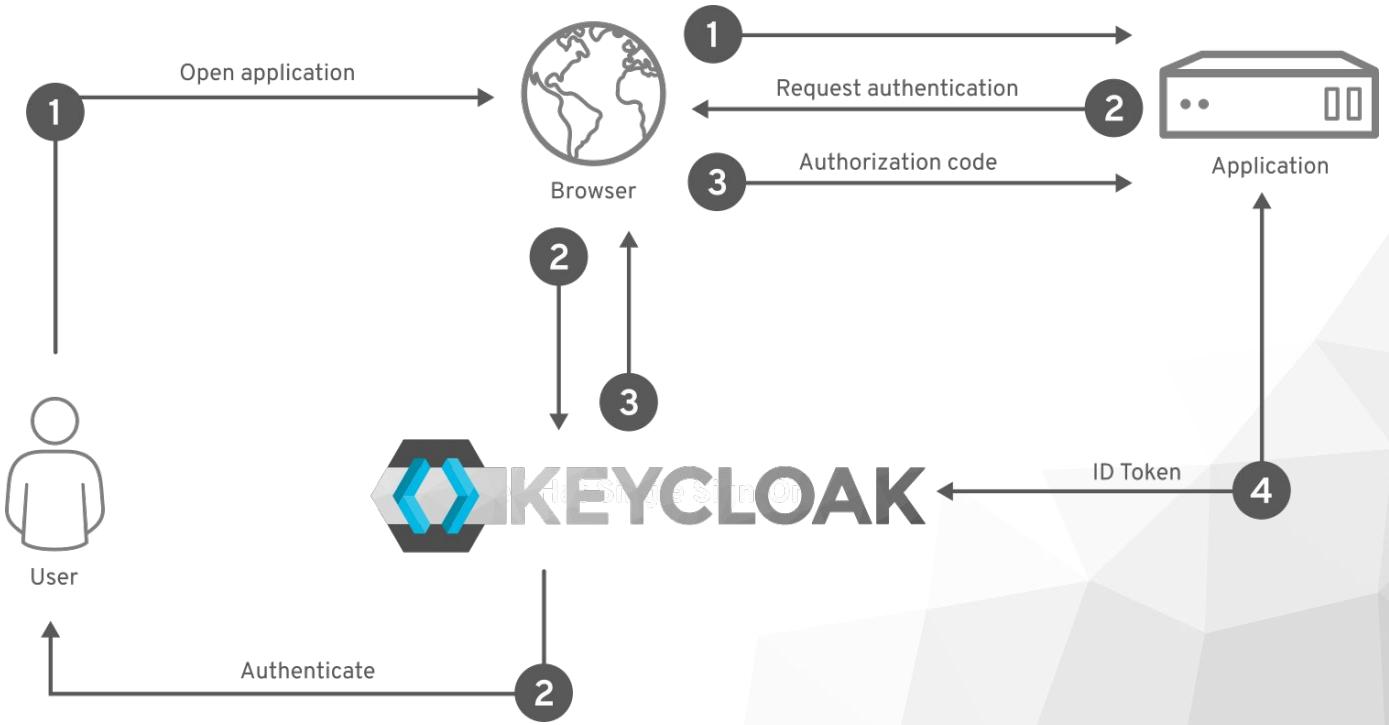
More cool features...

- Social brokering support (Facebook, Google, Twitter...)
- Any 3rd party identity provider brokering (OpenID Connect, SAML2, OAuth2)
- 3rd party user storage (LDAP, SSSD, custom...)
- Custom authentication mechanisms (TOTP, Kerberos, WebAuthn). Possibility to add more
- Themes - enhance Keycloak forms
- Events - monitoring (User events, admin events)

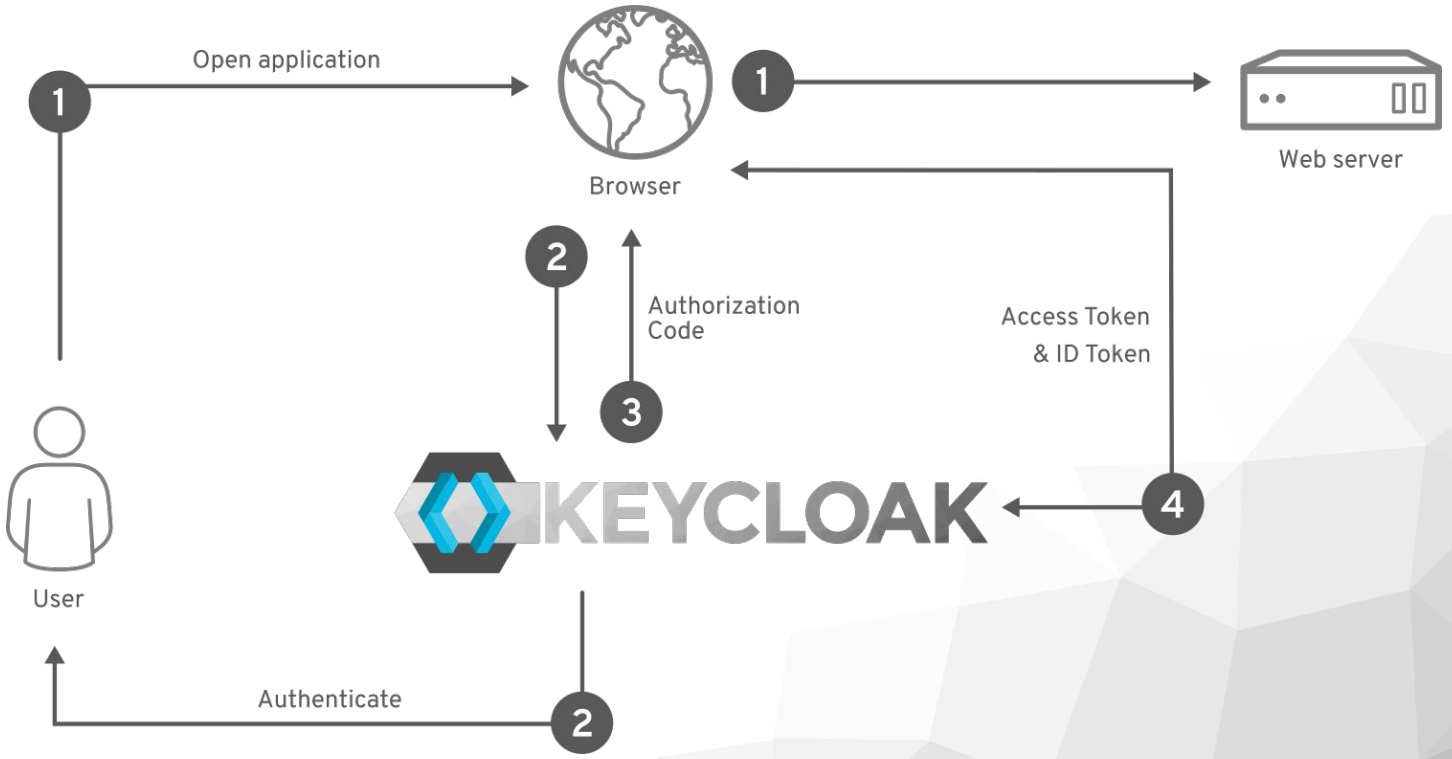
Securing your application

- 3rd party adapters to implement client side of the specification (Spring security, Wildfly, Quarkus)
- Almost no need to code anything in your app!
- Keycloak adapters (OIDC, SAML)

Monolithic application

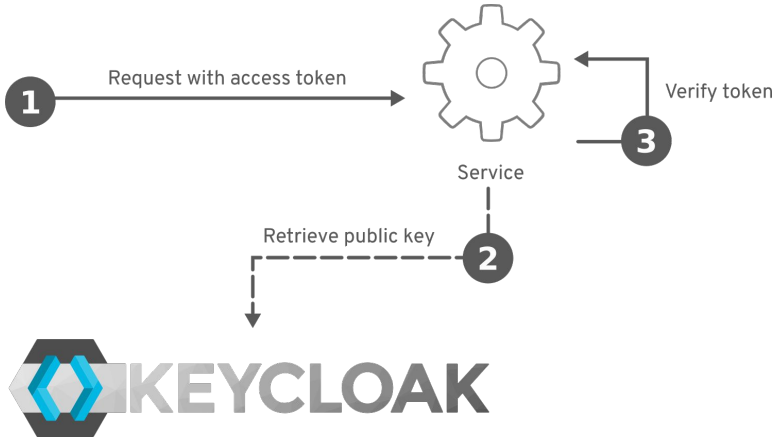


Single-page application

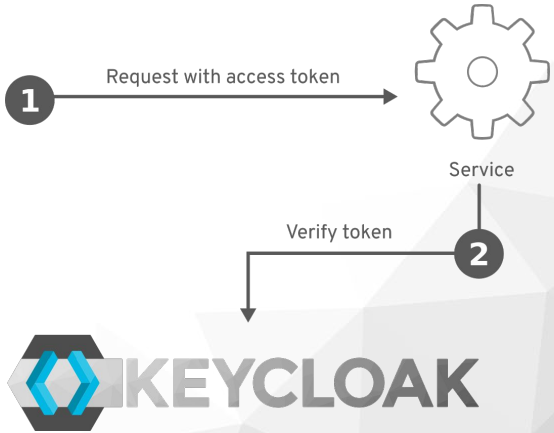


Service

Verify offline with signature



Verify online



Financial-grade API (FAPI)

- Set of rules to be used in the financial and banking applications
- The FAPI security profile can be applied to APIs in any market area that requires a higher level of security than provided by standard OAuth or OpenID Connect (OIDC).
- Specifies additional security requirements for all parties of OIDC/OAuth workflow (Authorization server (Keycloak), Client application, Service (resource server)).
- For Keycloak, especially OP requirements are important. Admins should make sure to implement requirements for RP and services (out of Keycloak responsibility)

FAPI 1

- 2 main specifications - “baseline” and “advanced”
- FAPI 1 baseline - It specifies a baseline security profile of OAuth that is suitable for protecting APIs with a moderate inherent risk.
- Requirements for key sizes, algorithms, etc. Most of them enforced in Keycloak by default
- Requires consent screen to be shown to the user during authentication
- Requires PKCE
- For confidential clients, requires secured client authentication mechanism based on JWT or MTLS. Simple clientSecret authentication not allowed

FAPI 1 - continue

- FAPI 1 advanced - specifies an advanced security profile of OAuth that is suitable to be used for protecting APIs with high inherent risk.
- More strict than “baseline”
- Protection against authorization request tampering and authorization response tampering (among others)
- Requires confidential clients with client authentication based on MTLS or private key JWT
- Requires authorization request to be sent with “request” parameter and/or PAR
- Requires authorization response to be protected (JWT or detached signature)
- Requires sender-constrained access tokens

FAPI - regional variants

- Various regions have their own flavours of the FAPI specification (usually based on the generic FAPI with additional requirements)

- Examples: OpenBanking Brasil, Australia CDR, OpenBanking UK

FAPI certifications

- OIDF provides certification testsuite, where implementations can certify themselves to prove the proper support of FAPI and OpenID Connect

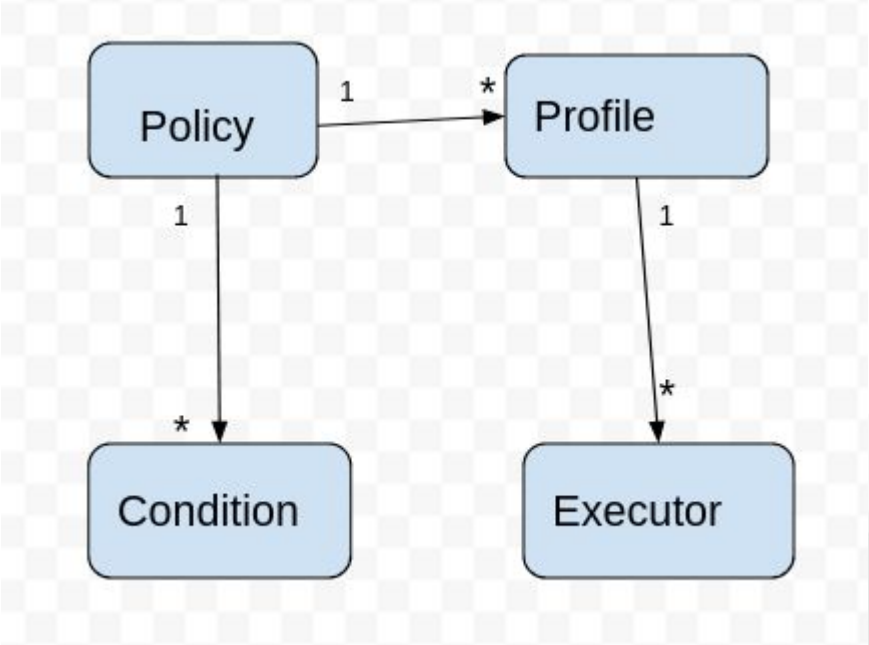
- Keycloak certified with FAPI Generic (Advanced profile) and FAPI CIBA also OpenBanking Brasil and Australia CDR variants

- Keycloak also certified as OpenID Connect provider (generic and logout profile)

Client policies

- Keycloak feature to for easier enforcement of profiles like FAPI
- Setup policies on what configuration client can have
- Enforce client behaviour in various events (OIDC requests, client registration/update etc)

Building blocks



Condition

- Specifies under which condition is the client policy used
- Has it's own SPI - ClientPolicyConditionProvider is provider class
- Also specifies when it is used (Client creation request, Token endpoint request etc) and by whom etc (ClientPolicyContext)

- Example conditions:
  - Evaluates if client created/updated by specific way (EG. OIDC registration, Admin REST)
  - Evaluates to true if client has specific access type (EG. public client)
  - Evaluates to true if specified "scope" is requested
  - Evaluates to true if creator/updator has specified role

Executor

- Do some validation action during specific action
  - EG. Checks if client uses secure PKCE method during authorization request
- Some executors can optionally do "augmentation" of the configuration during create/update client
  - EG. enforce the PKCE with the S256 method
  - Usually done by changing already existing switches (EG. "pkce.code.challenge.method" switch)
- Executor operates just on specific events
- Has it's own SPI - ClientPolicyExecutorProvider is provider class

Profile

- Used to make sure that client actions matches to the particular security profile (EG. FAPI, OpenBanking)

- Technically profile is list of executors

Policy

- Specifies which clients and when should comply with the particular profiles
- List of conditions and profiles. Profiles are executed if all applied conditions are YES ("and" logic enforced ATM)

Default client profiles for FAPI

- Keycloak has default client profiles for FAPI baseline, FAPI advanced and FAPI CIBA
- Administrator may need to configure policies with conditions to prescribe which clients should be FAPI compliant
- Example use-cases:
  - User “fapi-admin” can create only clients compliant with FAPI advanced
  - All realm confidential clients must conform to FAPI advanced
  - All realm public clients must conform to FAPI baseline

Demo

FAPI SIG

- Community special interest group
- Group of contributors interested in contribution of the FAPI and OpenID Connect related features
- Regular meetings (monthly) to discuss next steps, contributions in progress etc.
- Details: https://github.com/keycloak/kc-sig-fapi

Thanks

Keycloak website https://www.keycloak.org/

Keycloak source https://github.com/keycloak/keycloak

Keycloak documentation - FAPI and Client policies

Demo https://github.com/mposolda/fapi-demo

FAPI SIG https://github.com/keycloak/kc-sig-fapi

Mail mposolda@redhat.com