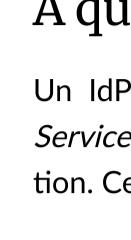


Sortie de Keycloak 7.0

Posté par Mathieu CLAUDEL (site web personnel) le 27 août 2019 à 14:31. Édité par 4 personnes. Modéré par Ysabeau. Licence CC By-SA.

Étiquettes : fédération, ldap, kerberos, authentification, otp, iam, identité



Keycloak est sorti le 24 août dans la [version 7.0](#).

Keycloak est un fournisseur d'identités (*Identity Provider* ou *IdP*) moderne, écrit en Java et compatible par défaut avec les protocoles de fédération d'identités [SAML v2](#) et [OpenID Connect \(OIDC\)](#)/OAuth2. Il est sous [licence Apache](#) et est porté par Red Hat.

À quoi ça sert

Un IdP permet à une application (souvent nommée *Service Provider* ou *SP*) de déléguer son authentification. Cela a, entre autres, plusieurs avantages :

- permettre aux développeurs de se concentrer sur les fonctionnalités métier en n'ayant pas à se préoccuper des aspects de sécurité liés à l'authentification, soit en intégrant directement une bibliothèque compatible avec un des deux protocoles, soit en utilisant un module sur le serveur Web ou encore un adaptateur Keycloak ([liste non exhaustive des possibilités](#)) ;
- centraliser l'authentification et donc permettre de faire de l'authentification unique (*Single Sign-On* ou *SSO*) ;
- unifier les méthodes d'authentifications et les faire évoluer sans modification des applications ;
- réinternaliser les authentifications des applications [SaaS](#) et ainsi maîtriser la prolifération des identités numériques ; la désactivation des comptes s'en trouve simplifiée (plus d'oubli de suppression d'un compte SaaS lors du départ d'un collaborateur).

Fonctionnalités principales

- *single sign-on* : authentification unique ;
- protocoles standards ;
- sécurisation des applications et service simplifié ;
- compatible LDAP comme référentiel d'utilisateurs externe ;
- délégation d'authentification (*social login*) ;
- haute performance : grappe de serveurs, « *scalable* », haute disponibilité ;
- pleinement compatible avec la conteneurisation ;
- thèmes simples à implémenter ;
- authentification forte par code à usage unique natif (OTP) via FreeOTP ou encore Google Authenticator ;
- auto-dépannage en cas d'oubli de mot de passe ;
- auto-création de comptes (par formulaire ou par authentifications dites sociales) ;
- [extensible](#) : base d'utilisateurs, méthodes d'authentification, protocoles.

Aller plus loin

[Keycloak](#) (355 clics)

[Téléchargement](#) (50 clics)

[Source](#) (61 clics)

[Débuter – Documentations](#) (132 clics)

(9 commentaires).

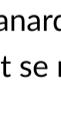
Markdown

EPUB

selfid

Posté par devnewton (site web personnel) le 28 août 2019 à 22:43. Évalué à 4.

J'ai travaillé avec Keycloak et OpenID Connect, mais je trouve ça trop compliqué.



Le problème avec les protocoles classiques est qu'il faut gérer des comptes utilisateurs, des login/mdp, des codes de confirmation par SMS ou mail...

D'où l'idée d'un protocole basé sur la confiance: selfid.

Le principe est simple:

1. l'utilisateur se connecte sur un site.
2. le site le redirige vers un IDP selfid.
3. l'IDP affiche un formulaire demandant à l'utilisateur de donner son nom, ses droits et de cocher deux cases: la première pour dire qu'il s'engage sur l'honneur à fournir des informations exactes, la seconde pour indiquer qu'il confirme être bien lui.

4. l'IDP forge un jeton JWT contenant le nom de l'utilisateur et ses droits puis le redirige vers le site en passant le jeton en paramètre.

Je réfléchis aussi à une future version, selfid coïncide, pour authentifier les canards, mais je ne sais pas si les anatides peuvent se reconnaître.

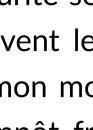
--

Le post ci-dessus est une grosse connerie, ne le lisez pas sérieusement.

[^] # Re: selfid

Posté par Mathieu CLAUDEL (site web personnel) le 29 août 2019 à 08:48. Évalué à 2.

Salut,



Le problème avec les protocoles classiques est qu'il faut gérer des comptes utilisateurs, des login/mdp, des codes de confirmation par SMS ou mail...

Je pense que tu lies protocoles (SAML v2, OpenID Connect,...) et Gestion des Identités.

On peut tout à fait faire ce que tu décris en utilisant l'un de ces protocoles (qui créer une relation de confiance entre IdP et SP et par extension aux identités fournies mais si elles sont farfelue). Par exemple en SAMLv2 tu peux utiliser <https://stuidp.sustainsys.com/> qui est un IdP compatible SAML v2 dans lequel tu forges toi-même le jeton de retour (assez intéressant pour les tests).

Pour faire une gestion d'identité simple pour des applications grand public (par opposition à une application interne à une entreprise où la maîtrise des salariés est indispensable), avec keycloak, tu peux activer la création de compte ce qui permet au utilisateur de renseigner eux même leur identité et de gérer son cycle de vie.

Je tiens aussi à souligner que sans méthodes d'authentifications (mot de passe, OTP, Certificat, Webauthn,...), l'IdP ne peut vérifier l'identité revendiquée par l'utilisateur et ouvre la porte à l'usurpation.

D'où l'idée d'un protocole basé sur la confiance: selfid.

Un protocole qui est basé sur la confiance est [OpenID](#) (pas connect !) mais il ne résout pas le problème de la création et du cycle de vie des identités :). Mais ça n'a pas spécialement pris.

Je réfléchis aussi à une future version, selfid coïncide, pour authentifier les canards, mais je ne sais pas si les anatides peuvent se reconnaître.

Voilà un beau projet de rfc pour le 1er avril prochain !

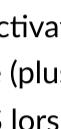
[^] # Re: selfid

Posté par devnewton (site web personnel) le 29 août 2019 à 10:00. Évalué à 3.

Pour faire une gestion d'identité simple pour des applications grand public (par opposition à une application interne à une entreprise où la maîtrise des salariés est indispensable), avec keycloak, tu peux activer la création de compte ce qui permet au utilisateur de renseigner eux même leur identité et de gérer son cycle de vie.



Je préférerais un OTP mail. Je ne vois pas l'intérêt de créer un compte pour certains sites (genre où tu te connectes une fois par an).



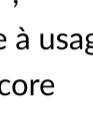
--

Le post ci-dessus est une grosse connerie, ne le lisez pas sérieusement.

[^] # Re: selfid

Posté par Mathieu CLAUDEL (site web personnel) le 29 août 2019 à 10:12. Évalué à 2.

Salut,



Le problème avec les protocoles classiques est qu'il faut gérer des comptes utilisateurs, des login/mdp, des codes de confirmation par SMS ou mail...

Je pense que tu lies protocoles (SAML v2, OpenID Connect,...) et Gestion des Identités.

On peut tout à fait faire ce que tu décris en utilisant l'un de ces protocoles (qui créer une relation de confiance entre IdP et SP et par extension aux identités fournies mais si elles sont farfelue). Par exemple en SAMLv2 tu peux utiliser <https://stuidp.sustainsys.com/> qui est un IdP compatible SAML v2 dans lequel tu forges toi-même le jeton de retour (assez intéressant pour les tests).

Pour faire une gestion d'identité simple pour des applications grand public (par opposition à une application interne à une entreprise où la maîtrise des salariés est indispensable), avec keycloak, tu peux activer la création de compte ce qui permet au utilisateur de renseigner eux même leur identité et de gérer son cycle de vie.

Je tiens aussi à souligner que sans méthodes d'authentifications (mot de passe, OTP, Certificat, Webauthn,...), l'IdP ne peut vérifier l'identité revendiquée par l'utilisateur et ouvre la porte à l'usurpation.

D'où l'idée d'un protocole basé sur la confiance: selfid.

Un protocole qui est basé sur la confiance est [OpenID](#) (pas connect !) mais il ne résout pas le problème de la création et du cycle de vie des identités :). Mais ça n'a pas spécialement pris.

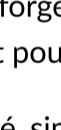
Je réfléchis aussi à une future version, selfid coïncide, pour authentifier les canards, mais je ne sais pas si les anatides peuvent se reconnaître.

Voilà un beau projet de rfc pour le 1er avril prochain !

[^] # Re: selfid

Posté par devnewton (site web personnel) le 29 août 2019 à 11:37. Évalué à 3.

Pour faire une gestion d'identité simple pour des applications grand public (par opposition à une application interne à une entreprise où la maîtrise des salariés est indispensable), avec keycloak, tu peux activer la création de compte ce qui permet au utilisateur de renseigner eux même leur identité et de gérer son cycle de vie.



Je préférerais un OTP mail. Je ne vois pas l'intérêt de créer un compte pour certains sites (genre où tu te connectes une fois par an).



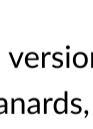
--

Le post ci-dessus est une grosse connerie, ne le lisez pas sérieusement.

[^] # Re: selfid

Posté par Mathieu CLAUDEL (site web personnel) le 29 août 2019 à 10:12. Évalué à 2.

Salut,



Le problème avec les protocoles classiques est qu'il faut gérer des comptes utilisateurs, des login/mdp, des codes de confirmation par SMS ou mail...

Je pense que tu lies protocoles (SAML v2, OpenID Connect,...) et Gestion des Identités.

On peut tout à fait faire ce que tu décris en utilisant l'un de ces protocoles (qui créer une relation de confiance entre IdP et SP et par extension aux identités fournies mais si elles sont farfelue). Par exemple en SAMLv2 tu peux utiliser <https://stuidp.sustainsys.com/> qui est un IdP compatible SAML v2 dans lequel tu forges toi-même le jeton de retour (assez intéressant pour les tests).

Pour faire une gestion d'identité simple pour des applications grand public (par opposition à une application interne à une entreprise où la maîtrise des salariés est indispensable), avec keycloak, tu peux activer la création de compte ce qui permet au utilisateur de renseigner eux même leur identité et de gérer son cycle de vie.

Je tiens aussi à souligner que sans méthodes d'authentifications (mot de passe, OTP, Certificat, Webauthn,...), l'IdP ne peut vérifier l'identité revendiquée par l'utilisateur et ouvre la porte à l'usurpation.

D'où l'idée d'un protocole basé sur la confiance: selfid.

Un protocole qui est basé sur la confiance est [OpenID](#) (pas connect !) mais il ne résout pas le problème de la création et du cycle de vie des identités :). Mais ça n'a pas spécialement pris.

Je réfléchis aussi à une future version, selfid coïncide, pour authentifier les canards, mais je ne sais pas si les anatides peuvent se reconnaître.

Voilà un beau projet de rfc pour le 1er avril prochain !

[^] # Re: selfid

Posté par devnewton (site web personnel) le 29 août 2019 à 11:37. Évalué à 3.

Pour faire une gestion d'identité simple pour des applications grand public (par opposition à une application interne à une entreprise où la maîtrise des salariés est indispensable), avec keycloak, tu peux activer la création de compte ce qui permet au utilisateur de renseigner eux même leur identité et de gérer son cycle de vie.

Je préférerais un OTP mail. Je ne vois pas l'intérêt de créer un compte pour certains sites (genre où tu te connectes une fois par an).

--

Le post ci-dessus est une grosse connerie, ne le lisez pas sérieusement.

[^] # Re: selfid

Posté par Mathieu CLAUDEL (site web personnel) le 29 août 2019 à 10:12. Évalué à 2.

Salut,

Le problème avec les protocoles classiques est qu'il faut gérer des comptes utilisateurs, des login/mdp, des codes de confirmation par SMS ou mail...

Je pense que tu lies protocoles (SAML v2, OpenID Connect,...) et Gestion des Identités.

On peut tout à fait faire ce que tu décris en utilisant l'un de ces protocoles (qui créer une relation de confiance entre IdP et SP et par extension aux identités fournies mais si elles sont farfelue). Par exemple en SAMLv2 tu peux utiliser <https://stuidp.sustainsys.com/> qui est un IdP compatible SAML v2 dans lequel tu forges toi-même le jeton de retour (assez intéressant pour les tests).

Pour faire une gestion d'identité simple pour des applications grand public (par opposition à une application interne à une entreprise où la maîtrise des salariés est indispensable), avec keycloak, tu peux activer la création de compte ce qui permet au utilisateur de renseigner eux même leur identité et de gérer son cycle de vie.

Je tiens aussi à souligner que sans méthodes d'authentifications (mot de passe, OTP, Certificat, Webauthn,...), l'IdP ne peut vérifier l'identité revendiquée par l'utilisateur et ouvre la porte à l'usurpation.

D'où l'idée d'un protocole basé sur la confiance: selfid.

Un protocole qui est basé sur la confiance est [OpenID](#) (pas connect !) mais il ne résout pas le problème de la création et du cycle de vie des identités :). Mais ça n'a pas spécialement pris.

Je réfléchis aussi à une future version, selfid coïncide, pour authentifier les canards, mais je ne sais pas si les anatides peuvent se reconnaître.

Voilà un beau projet de rfc pour le 1er avril prochain !

[^] # Re: selfid

Posté par devnewton (site web personnel) le 29 août 2019 à 11:37. Évalué à 3.

Pour faire une gestion d'identité simple pour des applications grand public (par opposition à une application interne à une entreprise où la maîtrise des salariés est indispensable), avec keycloak, tu peux activer la création de compte ce qui permet au utilisateur de renseigner eux même leur identité et de gérer son cycle de vie.

Je préférerais un OTP mail. Je ne vois pas l'intérêt de créer un compte pour certains sites (genre où tu te connectes une fois par an).

Mais tu veux dire créer un mot de passe ?



Un jeton plutôt (donc avec un périmètre et une durée de vie).

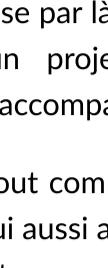
--

Le post ci-dessus est une grosse connerie, ne le lisez pas sérieusement.

[^] # Re: selfid

Posté par seveso le 30 août 2019 à 00:02. Évalué à 3.

Je préfèrerais un OTP mail



Est-ce que le projet [portier](#)

pourrait répondre à ton besoin ? Si oui alors je t'encourage à faire revivre ce projet :)

J'adorerais voir Portier en production mais personnellement je ne peux pas m'y investir tellement ça va bien au delà de mes capacités techniques et du temps libre que je pourrais y consacrer.

En tout cas si quelqu'un passe par là et a envie de faire du Rust sur un projet d'utilité publique... Tous mes vœux l'accompagne.

Pour information, Portier (tout comme Mozilla Persona son prédecesseur lui aussi abandonné) est une solution permettant une authentification sans mot de passe. Un peu comme les boutons "Facebook" ou "Google" mais sans GAFAM à l'intérieur. Et c'est auto-hébergable. Que du bonheur pour les personnes qui aiment leur vie privée.

[^] # Re: selfid

Posté par Joris Dedieu (site web personnel) le 31 août 2019 à 14:09. Évalué à 3. Dernière modification le 31 août 2019 à 14:09.

J'ai travaillé avec Keycloak et OpenID Connect, mais je trouve ça trop compliqué.



Par rapport à d'autres implémentations telles que CAS ou Shibboleth, je trouve Keycloak très simple au contraire. Les protocoles SAMLv2 et OpenID Connect, ne sont pas intuitifs par contre.

tache

Posté par Nicolas Boulay (site web personnel) le 29 août 2019 à 13:55. Évalué à 5.

Super produit mais je trouve qu'il faut un peu trop comprendre le protocole et les différents modes pour faire les choses correctement.



Savoir quoi mettre dans les header http devraient être un mode avancé.

Il faudrait un assistant pour ne pas avoir à choisir mode implicit ou pas. Mettre en place une authentification dans une simple appli spa + protection des api rest n'a rien d'évident. Alors que cela semble être la manière la plus classique de faire un nouveau wite web.

--

"La première sécurité est la liberté"

Avec un jour d'avance.¹

Posté par FantastIX le 29 août 2019 à 20:27. Évalué à 2. Dernière modification le 29 août 2019 à 20:28.

moderne, écrit en Java ...



-->[]

(¹ Ou deux jours de retard, c'est selon)

[Suivre le flux des commentaires](#)

Note : les commentaires appartiennent à celles et ceux qui les ont postés. Nous n'en sommes pas responsables.

[Revenir en haut de page](#)

Derniers commentaires

Étiquettes (tags) populaires

Sites amis

À propos de LinuxFr.org

- Re: Helene, je m'app...
- Helene, je m'appelle...
- Re: essai par rapport...
- Re: C'est la faute de ...
- Re: Irresponsabilité ...
- Re: Zorin = Ubuntu
- yet another rc mana...
- Re: Bon débarras ?
- Re: Zorin = Ubuntu
- Re: MOOC Program...
- Re: On pousse en in...
- Re: Mauvaise versio...

- intelligence_artificielle
- merdification
- grands_modèles_de...
- hppa
- états-unis
- administration_fran...
- sortie_version
- donald_trump
- capitalisme
- capitalisme_de_surv...
- linux
- note_de_lecture

- April
- Agenda du Libre
- FramaSoft
- Éditions D-BookeR
- Éditions Eyrolles
- Éditions Diamond
- Éditions ENI
- La Quadrature du Net
- Lea-Linux
- En Vente Libre
- Grafik Plus
- Open Source Initiative

- Mentions légales
- Faire un don
- L'équipe de LinuxFr....
- Informations sur le s...
- Aide / Foire aux que...
- Suivi des suggestion...
- Règles de modération
- Statistiques
- API pour le développ...
- Code source du site
- Plan du site

[Suivre le flux des commentaires](#)