Emmanuel Decoux
Xavier Montagutelli

LemonLDAP::NG à l'Université de Limoges

Un Web SSO et Portail avec contrôle d'accès utilisé comme serveur CAS et OpenID Connect

université ouverte
source de réussites

Université de Limoges

Remerciements et crédits

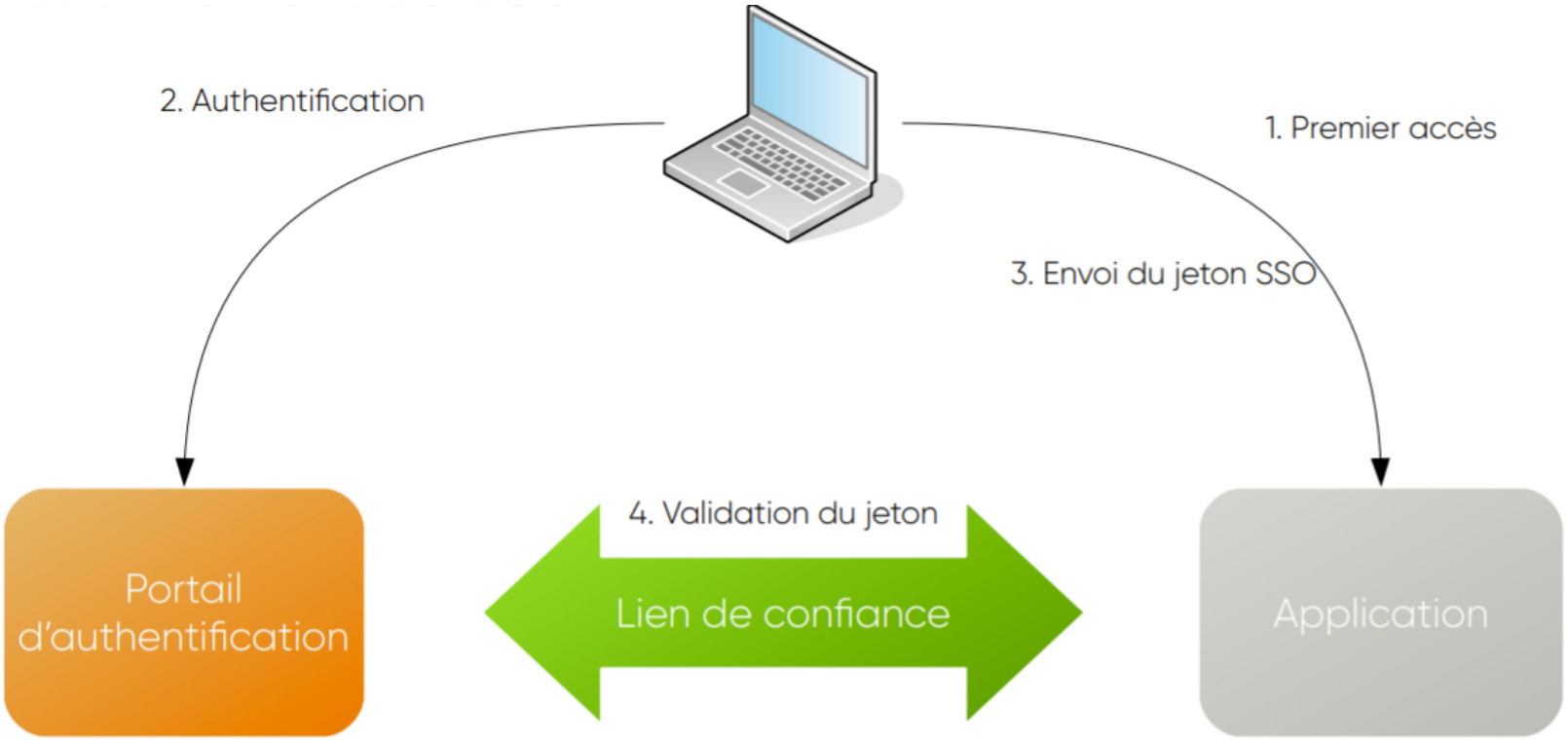
- Clément Oudot, conférence LDAPCon 2019 (Slides)

A savoir

- Licence GPL
- Site https://lemonldap-ng.org
- Membre de OW2
- Partenaire FranceConnect
- Ecrit en Perl et Javascript
- Utilisé par la Gendarmerie Nationale (170.000 utilisateurs)

LL::NG – Un SSO pour applications webs

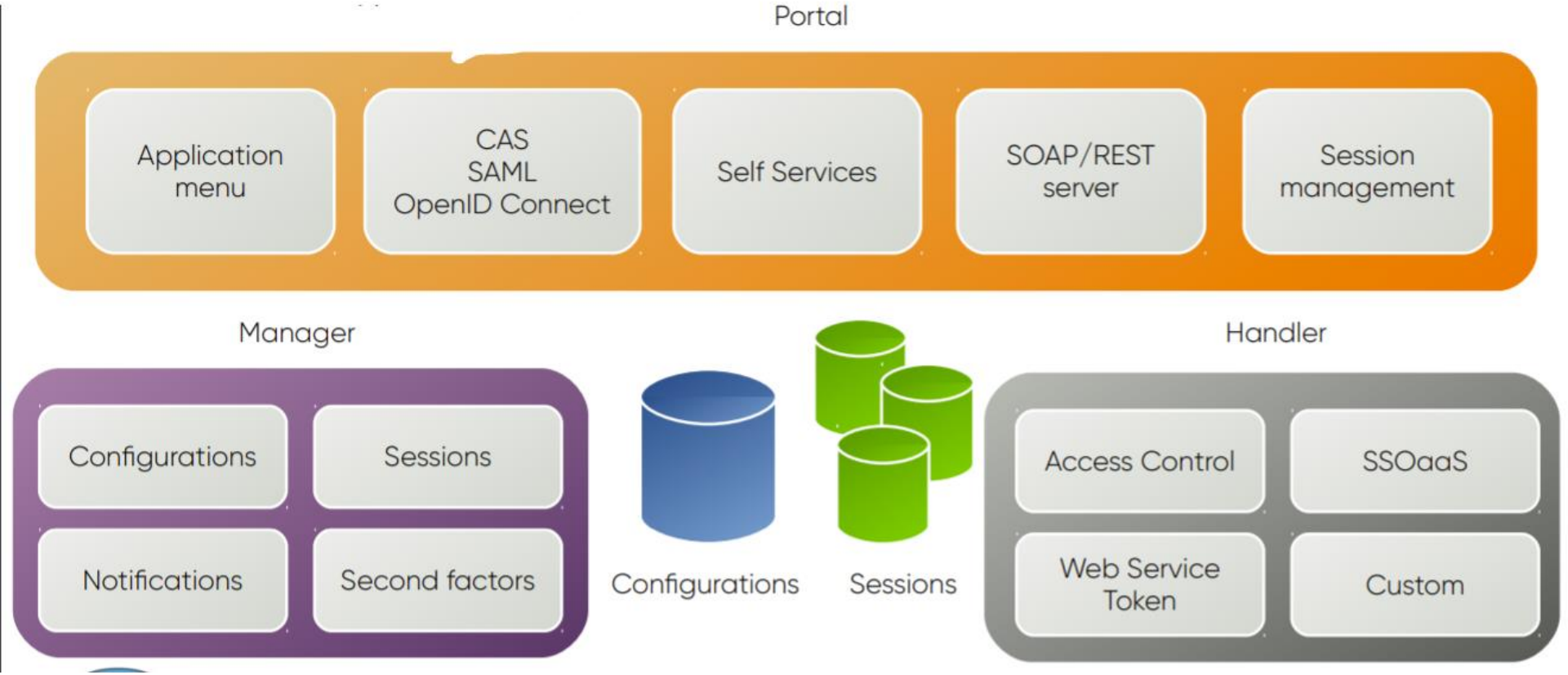
En simplifié :



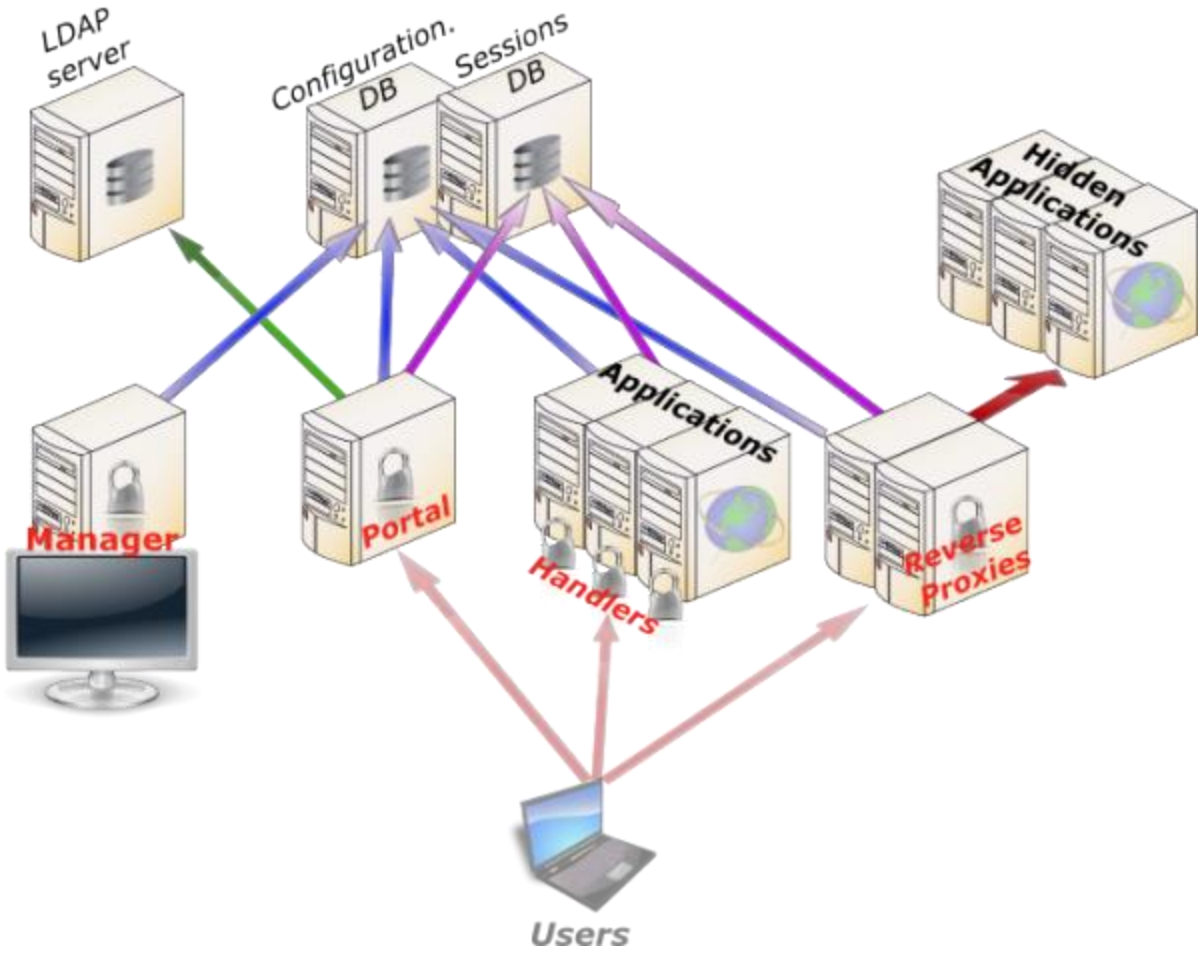
Principales fonctionnalités

- Authentification unique (WebSSO)
  - Chaînage et choix des modules d'authentification
  - Gestion du mot de passe, création de compte
  - Authentification multi-facteurs
- Contrôle d'accès aux applications et gestion des autorisations
- Portail d'applications (affiche une liste des applications accessibles à l'utilisateur connecté)
- Protection des applications Web et des API/WebServices (route, URL)

LL::NG – Composants



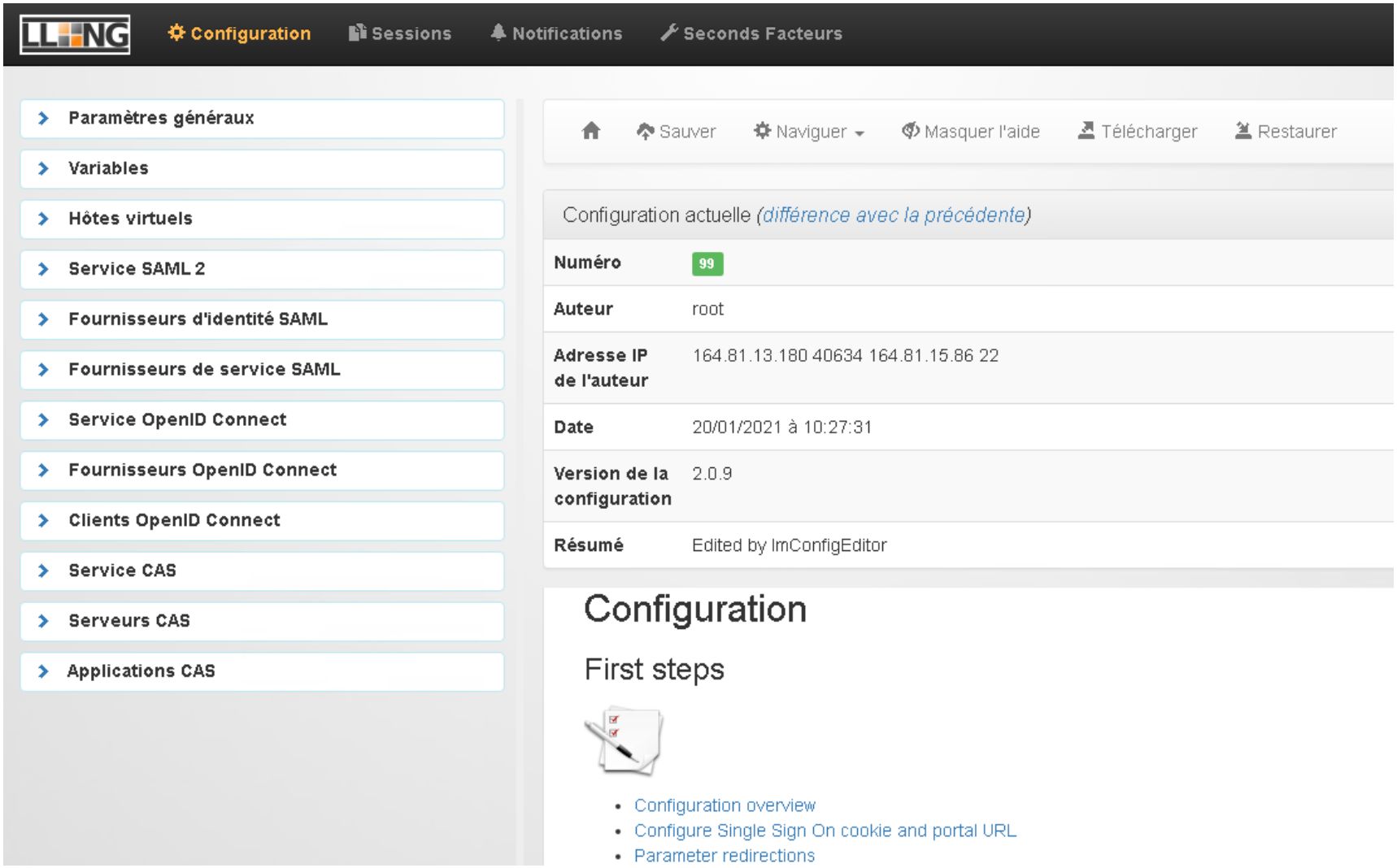
LL::NG – Composants (2)



LL::NG – Manager

- La partie *Manager* … sert à gérer LL::NG
  - Pour le paramétrer
  - Pour lister les sessions actives, ou les terminer
  - Pour créer des *notifications* (des messages qui s'affichent une seule fois lors de la connexion d'un usager)
- La gestion peut aussi se faire en CLI
- Ou par API REST

LL::NG – Manager (2)



LL::NG – Portail

- Le *Portail* est le coeur de LL::NG
  - Il gère l'authentification :
    - par login/mdp
    - Ou par authentification gérée via le serveur web
    - Ou par un fournisseur d'identité externe
    - Les modules sont chaînables, et/ou sélectionnables par des règles
    - Gère aussi la partie 2FA (U2F, TOTP, Yubikey, e-mail, etc.)
  - Sert de *fournisseur d'identité* : CAS, SAML, OpenID Connect
    - Peut donc faire *proxy* entre 2 systèmes d'identités différents
  - Sert de serveur SOAP ou REST : permet à une application d'inter-agir avec LL::NG par API
  - Permet la gestion des comptes utilisateurs :
    - Changement de mot de passe, création de compte, recouvrement mdp
    - Affichage de l'historique des connexions
    - Affichage des consentements OIDC
  - Affiche un portail des applications accessibles à l'utilisateur
  - Affiche les notifications aux utilisateurs

LL::NG – Portail, authentification, utilisateur

- Plusieurs méthodes (modules) d'authentifications disponibles
  - LDAP, AD, CAS, BDD, FaceBook, LinkedIn, OIdC (google, France Connect), Radius, Rest, SAML, PAM, Kerberos, …
  - Une *combinaison* de modules est possibles (plusieurs modules déclarés, et une règle de sélection appliquée)
- A l'UL : LDAP
  - Vérification « traditionnelle » du *login* / *mdp*
  - A noter, compatible avec le draft *password policy* (forcer le changement du mdp, vérifier la complexité, tenir compte des comptes verrouillés)
  - Chaînage Kerberos (via AD) + LDAP en phase de test
    - Nécessite de paramétrer les navigateurs pour autoriser Kerberos
- Après la phase d'authentification, un autre module (ou le même) récupère des données sur l'utilisateur authentifié
  - Le module LDAP permet de récupérer des attributs et les groupes

LL::NG – Authentification et session

- LL::NG crée une *session*
- Stockée dans la *base de données* des sessions
  - La configuration permet de choisir la base voulue
  - Partageable si on installe LL::NG sur plusieurs serveurs
- La session contient des *variables*
  - Par exemple, si on utilise le module LDAP pour les utilisateurs, les *attributs* peuvent être *exportés* dans la session
  - On peut aussi créer des variables (*macro*)
    - Par des formules (perl) simples dans la configuration de LL::NG
    - Par l'écriture de modules perl (*Custom Function*) et appel des fonctions dans le calcul d'une macro

LL::NG – Module de gestion des mots de passe

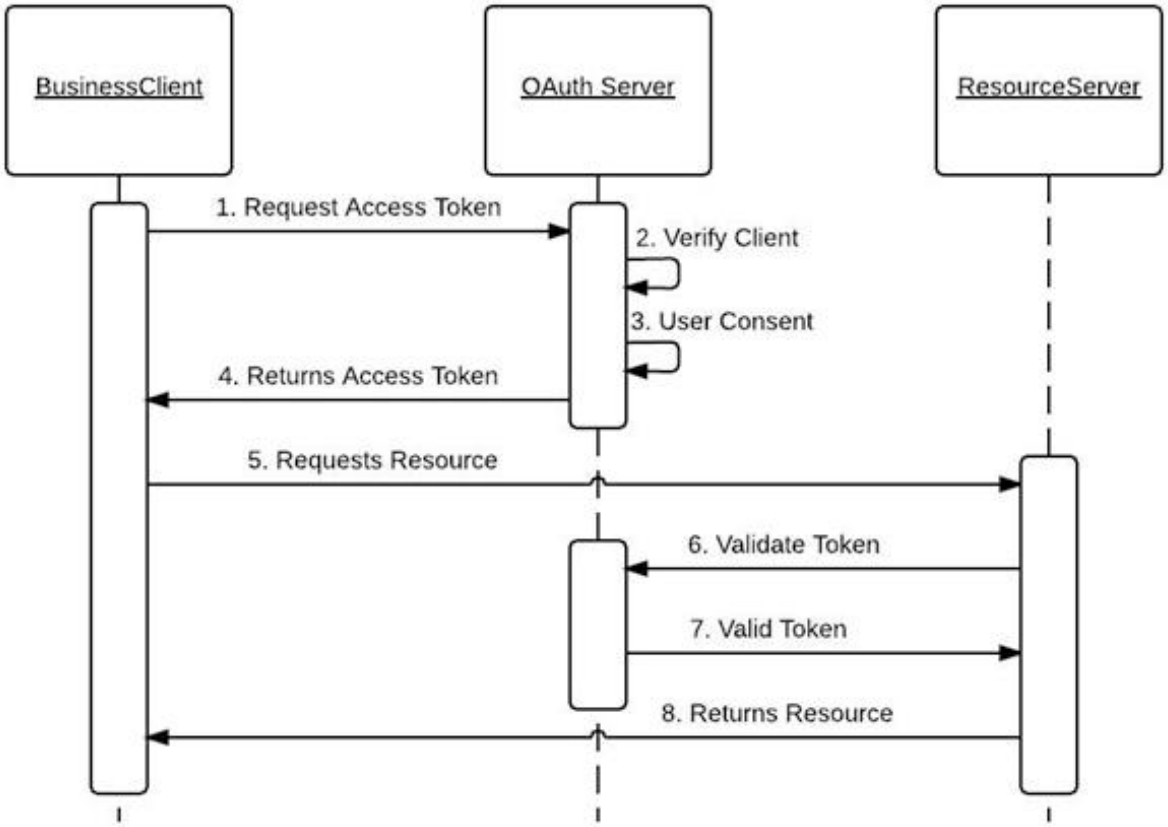
- On peut aussi activer un module pour la gestion des mots de passe
  - Pour changer son mot de passe, le recouvrer

LL::NG – Protocoles d'authentification bien connus

- LL::NG parle les protocoles :
  - CAS
  - SAML
  - OpenID Connect (OIDC)
- LL::NG peut s'en servir
  - En *module d'authentification* (pour vérifier une identité)
  - Et aussi comme *fournisseur d'identité* (pour qu'un client vérifie auprès de LL::NG une identité)

OpenID Connect (OIDC)

OpenID Connect est basé sur REST, OAuth 2.0 and JOSE stacks
cf : http://openid.net/connect/



LL::NG en Client OIDC

LL::NG peut agir en tant que client OIDC pour de multiples fournisseurs d'identité.

Il peut récupérer une identité via un token ID et les données rattachées via un point d'accès spécifique.

En tant que client OIDC, LL::NG supporte :
- Authorization Code flow
- Automatic download of JWKS
- JWT signature verification
- Access Token Hash verification
- ID Token validation
- Get UserInfo as JSON or as JWT
- Logout on EndSession end point

Il est possible de se connecter à des fournisseurs d'identités comme :
- Google
- FranceConnect

LL::NG en Fournisseur d'identité OIDC

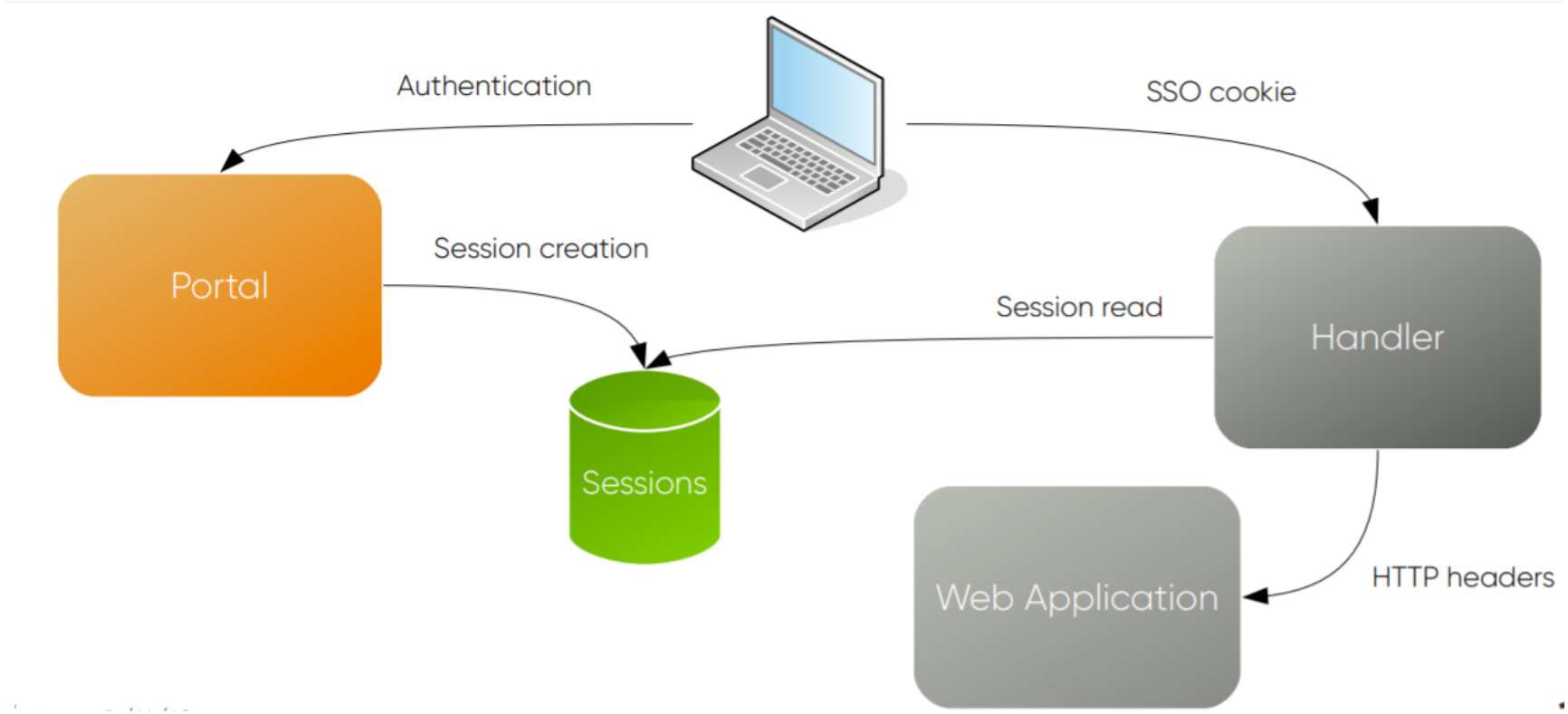
LL::NG peut agir en tant que fournisseur d'identité.
Il réponds aux requêtes OIDC en renvoyant une identité via un token ID et les données rattachées via un point d'accès spécifique.

En tant que fournisseur d'identité OIDC, LL::NG supporte :
- Authorization Code, Implicit and Hybrid flows
- Publication of JSON metadata and JWKS data (Discovery)
- prompt, display, ui_locales, max_age parameters
- Extra claims definition
- Authentication context Class References (ACR)
- Nonce
- Dynamic registration
- Access Token Hash generation
- ID Token signature (HS256/HS384/HS512/RS256/RS384/RS512)
- UserInfo endpoint, as JSON or as JWT
- Request and Request URI
- Session management
- FrontChannel Logout
- BackChannel Logout
- PKCE (Since 2.0.4) - See RFC 7636
- Introspection endpoint (Since 2.0.6) - See RFC 7662
- Offline access (Since 2.0.7)
- Refresh Tokens (Since 2.0.7)

LL::NG – Protection d'applications

- LL::NG peut « protéger » une application de plusieurs manières différentes
- Avec le « Handler »
  - Le composant s'intègre au niveau du serveur web (Apache / NGINX) dans le Virtual Host correspondant à la ressource à protéger
  - Le Vhost peut aussi faire reverse proxy
  - Il réalise, un peu comme un module d'authentification classique Apache ou NGINX, l'authentification et l'autorisation
  - L'application reçoit le « login » de l'utilisateur
    - Via la variable REMOTE_USER
    - Et avec d'autres infos via des headers HTTP
  - Pas utilisé à l'UL

LL::NG – Protection d'applications par handler



LL::NG – Protection d'applications (3)

- La protection d'application peut aussi reposer sur les protocoles d'authentification utilisés par les applis
- LL::NG est un fournisseur d'identité :
  - CAS
  - SAML
  - OpenID Connect

LL::NG – Autorisations

- On peut écrire des règles d'accès aux applications protégées
- Idem pour les services CAS (mais si on utilise cette fonctionnalité, tous les services autorisés doivent être déclarés)
- En utilisant des *conditions*
  - Code perl simple du type `$uid eq 'toto'`
  - Ou en écrivant des modules et en faisant appel aux méthodes dans les conditions
- Les variables de session sont utilisables

LL::NG – Comment ça s'installe ?

- Tarball
- Paquets Debian ou RPM ou Suse
- Docker

LL::NG – Base de données, choix important

- Plusieurs bases de données sont utilisées
  - Pour la configuration
  - Pour les *sessions*
  - Pour les *sessions persistantes* (stockage de l'historique des connexions, des notifications acceptées, des consentements OIdC)
- Plusieurs *backends* de stockage sont possibles : fichier, LDAP, REDIS, base de données, ...

LL::NG – Principales difficultés

- Une documentation pas facile en première lecture
- Une tournure très *perl-ish* dans les explications (culture perl bienvenue)

Custom Plugin : La force en souplesse

LL::NG permet d'intégrer des "Custom Plugin" en Perl.

Il est possible de :

- Rajouter du code "maison" sur des points d'entrée prévus.

Les processus d'authentifications des différents protocoles possèdes des hooks sur l'ensemble de étapes.

- Mais aussi de sur définir le code interne.

L'architecture de LL::NG utilise Mouse (dérivé de Moose) et permet facilement d'insérer du code maison avant, autour et/ou après les appels des méthodes internes.

Il suffit de déclarer le ou les plugins dans le manager, et de déposer les fichiers sources dans les dossiers d'inclusion de sources.

Notifications

LL::NG contient un plugin Notification pouvant être activé.

Une notification est définit soit pour un identifiant d'utilisateur précis, soit pour tous les utilisateurs, mais dans ce cas un filtre permet de cibler une population précise.

- Ce plugin permet de créer des notifications aux usagers apparaissant entre la fin de l'authentification et l'accès à la ressource demandée.
- Elle n'est affichée qu'une seule fois par utilisateur.
- Elle peut contenir une ou des case à cocher. Dans ce cas l'utilisateur doit obligatoirement les cocher pour poursuivre sa navigation.
- En tant qu'utilisateur, il est possible de lister l'ensemble des notifications déjà acceptées.

Les notifications ciblées

- Le filtre écrit en perl se base sur les variables de session

Merci

Imagine there are no passwords
Or maybe just only one
A single secured form
To access our applications
Imagine all the users
Loving security

You may say
I'm a hacker
But I'm not the only one
I hope one day
You will log in
Using the Single Sign On

Imagine applications
No more storing passwords
Relying on a token
Even for authorizations
Imagine all developers
Loving security

Imagine some protocols
Made by clever people
CAS, OpenID or SAML
Even WS Federation
Imagine authentication
Interoperability

John Lennon

https://ldapcon.org/2017/wp-content/uploads/2017/08/16_Cl%C3%A9ment-Oudot_PRE_LDAPCon2017_SSO-1.pdf