LemonLDAP::NG - A Full AAA Free Open Source WebSSO Solution

Christophe Maudoux, Selma Boumerdassi

LemonLDAP::NG

A Full AAA Free Open Source WebSSO Solution

Christophe Maudoux & Selma Boumerdassi
Cnam/Cedric Lab – Networks & IoT Research Team

Contact Information:
Cedric Lab/ROC Team
Conservatoire National des Arts & Métiers
2 rue Conté 75003 PARIS

Email: christophe.maudoux@cnam.fr
Website: https://lemonldap-ng.org

Abstract

Single Sign-On (SSO) is an authentication scheme that permits a user to log in with a single ID to any of several related, yet independent, systems. It allows the user to log in once and access services without re-entering authentication credentials. SSO solutions are classified depending on *Authentication*, *Authorization*, and *Accounting* features. AAA is an acronym defining a framework for intelligently controlling access to resources, enforcing security policies, auditing usage, and providing the information necessary to bill for services. These combined processes are considered *important* for effective network *management* and *security*.

Presentation

LemonLDAP::NG (LL::NG) is full AAA. It affords different ways for *authenticating* users, checks *authorizations* before accessing resources, and provides *logs*. *(i)* It relies on *back-ends* to store configuration and sessions. *(ii)* The *Portal* displays authentication screen, applications menu, and implements standard protocols. *(iii)* The *Manager* is the administration interface. *(iv)* The *Handler* protects applications working with HTTP headers.

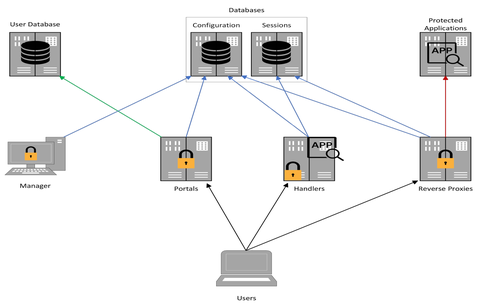

Figure 1: Main components

LL::NG is mainly coded in Perl and JavaScript by the core development team members: XAVIER GUIMARD (STSISI), CHRISTOPHE MAUDOUX (STSISI & Cnam), CLÉMENT OUDOT (Worteks) and MAXIME BESSON (Worteks).

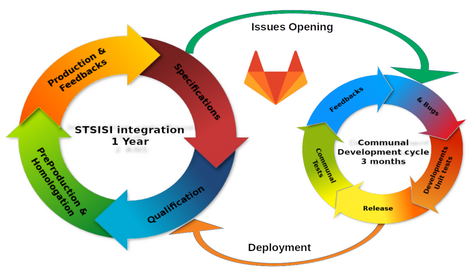

Figure 2: Development cycle

Authentication Methods & Protocols

LL::NG provides several authentication methods and implements standard protocols: **Protocols** GPG, Kerberos, AD, LDAP, CAS, SAMLv2, OIDC, Radius. **Connectors** Apache, Google, GitHub, PAM **Networks** Facebook, LinkedIn, Twitter

Combo back-ends like *Authentication choice* and *Combination* of authentication schemes can be employed.

Multi Factor Authentication (MFA) is also supported by LL::NG. It is a method to confirm a user's claimed identity by using a combination of two different factors between:

- something you know (login/password, ...)
- something you have (U2F Key, TOTP, smartphone, mail, ...)
- something you are (biometrics like fingerprints, ...)

Applications Protection

- SAMLv2 protocol does not require a direct link between SP and IdP. All network exchanges go through the browser.

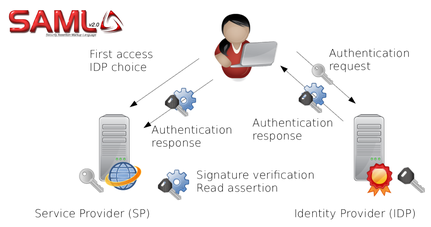

Figure 3: SAMLv2/Shibboleth kinematic

- OIDC standard defines 3 authentication flows but the *Hybrid* and *Implicit* flows are *deprecated* and should not be employed.

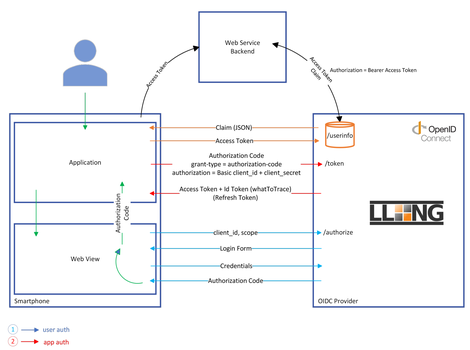

Figure 4: OIDC Authorization Code Flow kinematic

LL::NG can act as IdP/SP (SAML) or OP/RP (OIDC) and could be used as proxy between all these protocols!

- Handler code can be embedded by the protected applications but it requires to have access to the application code and to modify it. To avoid this, applications can be hidden behind Reverse Proxies that embed the Handler. So, protected applications are not exposed to users and directly requested (fig. 1).

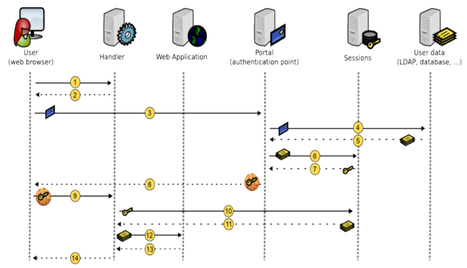

Figure 5: Handler kinematic

1. User tries to access a protected application, his request is caught by Handler 2. SSO cookie is not detected, so Handler redirects user to Portal 3. User authenticates on Portal 4. Portal checks authentication 5. If authentication succeeds, Portal collects user data 6. Portal creates a session to store user data 7. Portal gets the session Id 8. Portal creates a SSO cookie with the session Id as value 9. User is redirected to the protected application with a SSO cookie 10. Handler gets session Id from the cookie and retrieves user session data 11. Handler stores user data in its cache 12. Handler checks access rules and sends HTTP headers to the protected application 13. Protected application sends response to Handler 14. Handler forwards the response to user.

Specific Features & Corresponding Plugins

AdaptativeAuthLevel A user reaches an authentication level depending on the selected authentication module, and eventually MFA. This plug-in adapts the authLevel depending on other conditions, like network, device, ...

BruteForceProtection To prevent brute force attack

CheckDevOps Used for validating 'rules.json' files (fig. 7)

CheckUser To check session attributes, access rights and transmitted headers for a specific user and URL

ContextSwitching *Admins* can switch context other user

CrowdSec A free and open-source security automation tool leveraging local IP behaviour detection and a community-powered IP reputation system

Impersonation Users can assume identity of another user

Notification To notify some messages to users when log in

REST/SOAP To provide corresponding services

Advanced Usages

- The *ServiceToken* (ST) handler is a mechanism implemented to protect the *Server2Server exchanges*. A web application (WebApp1) may need to request some other web applications (WebAppN) on behalf of the authenticated users. You just have to provide a header containing the ST to WebApp1. Then WebApp1 can request WebAppN by appending the ST in the 'X-LLNG-TOKEN' header that will be caught by the handler.

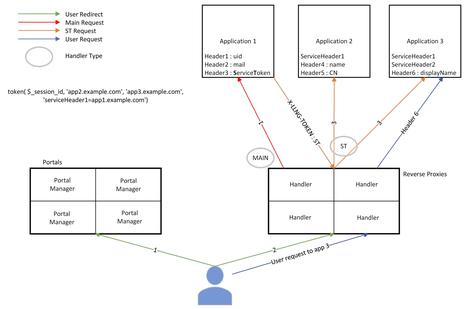

Figure 6: ServiceToken handler

- The *DevOps* handler is a mechanism to implement *SSO as a Service*. This handler is designed to retrieve VHost configuration from the website itself, not from LL:NG configuration. Rules and headers are set in a 'rules.json' file stored at the website root directory. The 'rules.json' file syntax and configuration can be checked by using the CheckDevOps plug-in.

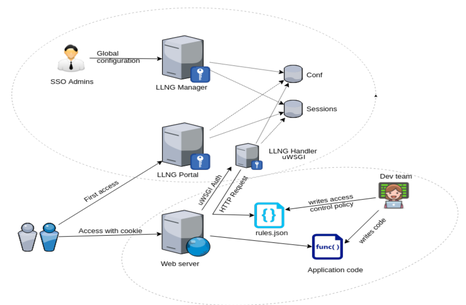

Figure 7: DevOps handler

Acknowledgements

Many thanks to ALEXANDRE KARIM from ESIEE Paris and the LemonLDAP::NG team for the figures and schemas.