Perspectives de la gestion d'identité par les technologies Web

XII^{ieme} Rencontres Mondiales du Logiciel Libre, Strasbourg
13 juillet 2011

Ates Mikaël, Dauvergne Benjamin
Entr'ouvert

Sommaire

Les fonctions de la gestion d'identité

- Authentification
- Contrôle d'accès
- Auditing/Reporting

Eclatement des fonctions de la GI

- Implémentations de logiques redondantes
- Redondance des fonctions d'administration
- Difficulté pour implémenter une politique de sécurité globale

Centralisation des fonctions de la GI

Objectifs

- Rationaliser les processus de GI (description, formalisation, analyse, optimisation, etc.)
- Réduire les implémentations redondantes : extraction des logiques redondantes (authentification, prise de décisions, etc.)
- Centraliser l'administration : diminution des interfaces
- Simplifier la mise en oeuvre d'une politique de sécurité globale (verifier la cohérence de l'ensemble)

Centralisation des fonctions de la GI

Bénéfices globaux

- Augmente potentiellement la sécurité du SI
- Réduction des coûts

Difficultés

- Externalisation des fonctions de GI des applications existantes
- Fournir des interface d'administration ergonomiques/adaptées aux applications

Les perspectives de la GI par les technologies Web

- Centralisation des services de GI dans les SI

Externalisation des fonctions de la GI

- La centralisation permet d'externaliser l'hébergement d'une partie des services de GI.
- La GI dans le "Cloud" ? Architectures Web distribuées où des services sont externalisés auprès d'un tiers.

Condition : interopérabilité des systèmes

- Besoin de protocoles de GI standards
- Besoin d'espaces de nom communs

Les perspectives de la GI par les technologies Web

- Services de GI “dans le cloud”

Les enjeux

- Contrôle des données personnelles diffusées, existantes et générées par l'usage des services en ligne
- Contrôle de la délégation d'accès à ses services personnels

En pratique

- Permettre à l'utilisateur de centraliser une partie de ses données personnelles
- Autorisations données par l'usager à un tiers pour l'accès à ses donnés personnelles ou à des services personnels
- Architecture pour la mise en oeuvre des autorisations et la gestion centralisée des données personnelles

Les perspectives de la GI

- Libération des données personnelles, externalisation des données personnelles et délégation de services
- Les délégations d'accès aux services font leur apparition (ex : twitter@anywhere)

A venir ?

- Extraction des données type carnet d'adresse, réputation de vendeur, etc.
- Migration vers des réseaux sociaux décentralisés
- Migration vers des services s'appuyant sur des données personnelles externalisées

Les enjeux

Exploitation de ses données d'identités issus de tiers multiples pour obtenir un accès auprès de tiers : contrôle d'accès basé sur la confiance

En pratique

- Certification de l'identité et des attributs d'identité
- Architecture pour l'échange de politique de contrôle d'accès et le transport/présentation des certificats

Les perspectives de la GI

- Insérer dans l'architecture un environnement de la confiance de l'usager hébergeant des services de la GI : pilotage d'une partie des échanges, reporting, autorisation, hébergement de données personnelles, certificats, et métadonnées, etc.
- Réaliser une architecture centrée sur l'utilisateur (user-centric)
- Standardisation d'un patchwork protocolaire et d'espaces de noms

Architecture de confiance globale

- Permettre aux fournisseurs de services d'établir des liens de confiance, potentiellement dynamiquement, envers les tiers sources d'informations d'identité
- En parallèle, découverte des clés de signature et des points d'entrée applicatifs
- Les architectures de confiance (CoT) déployées : fédérations universitaires, telco, etc.
- Puis interconnexion des fédérations (CoCoT)
- Besoin de mettre en oeuvre des critères communs pour déterminer la confiance à accorder : comment le fournisseur d'identité a établi l'identité, quelle politique de conservation des données personnelles, etc.

Architecture de confiance globale

Exemple des travaux de l'IAF (Identity Assurance Framework) de Kantara

- Définition d'une architecture batîe autour de la certifications d'organisations mettant des services en ligne (ex : Les fournisseurs de certificats)
- Certification pour un niveau d'assurance (LOA) : respect de divers critères (protocoles, gestion des données personnelles, etc.)
- Les LOA sont calqués sur ceux du NIST SP800-63
- Principe de l'assesseur (assessor) : tiers certificateur
- Les assesseurs sont eux-même certifiées
- Kantara opèrerait au démarrage la certification des assesseurs

The Actors in Kantara's Identity Assurance Framework:

KANTARA INITIATIVE IAF

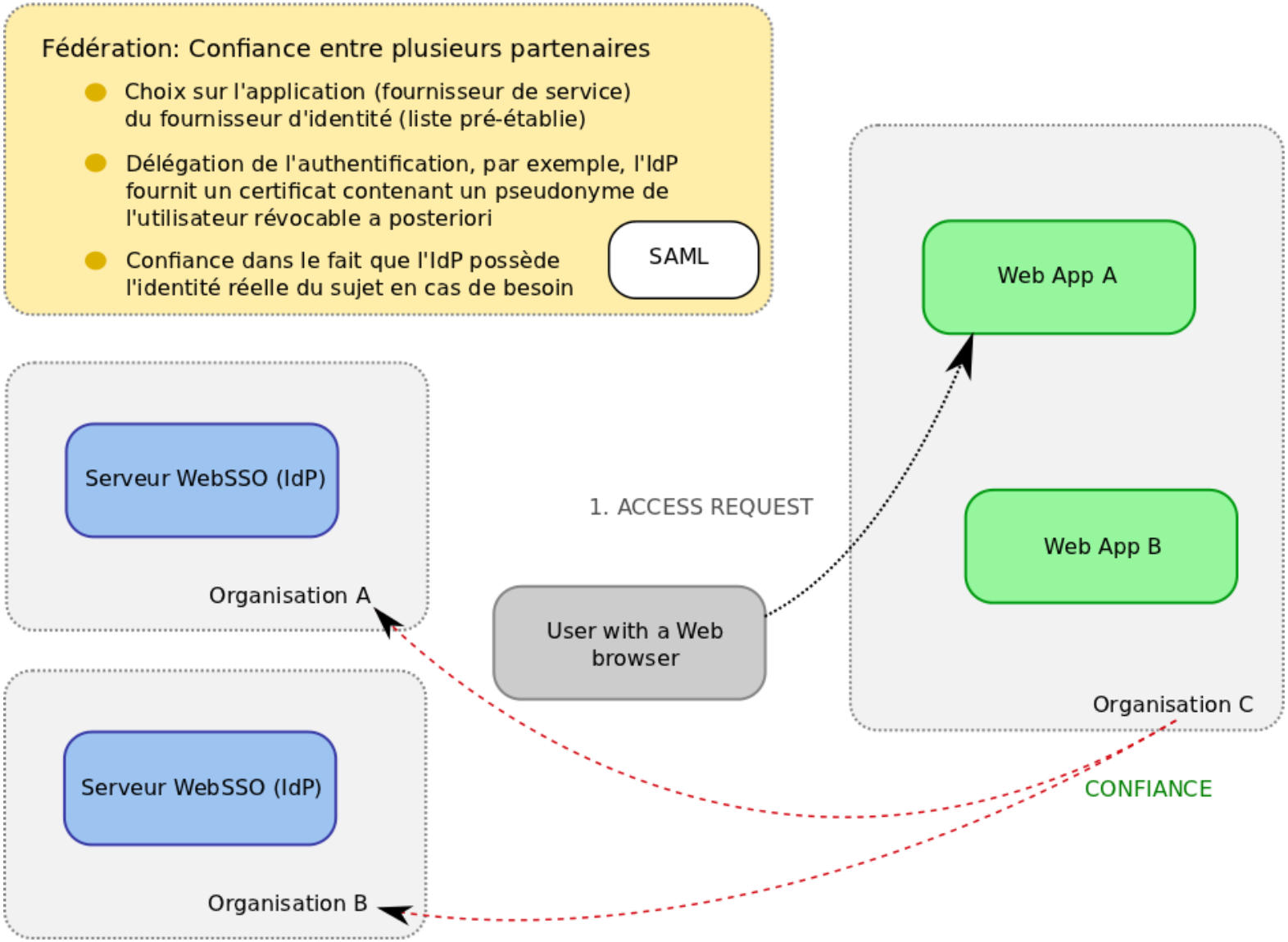
ASSESSORS	CREDENTIAL SERVICE PROVIDERS	FEDERATION OPERATORS	IDENTITY ASSURANCE FRAMEWORK 2.0 MAP
Get accredited by Kantara as an assessor against the IAF 1400 AQR Leverage demonstrable competencies to expedite certification	Get assessed by a Kantara Accredited Assessor for IAF certification against the IAF 1500 SACs Submit certification application to Kantara's Assurance Review Board (ARB) Obtain and maintain compliance and certification	Define criteria for identity assurance for their federation Map policy against IAF SAC and IAF profiles Accept or recommend IAF certification to its constituents	**NON-NORMATIVE**: • **(IAF 1000)** Overview • **(IAF 1100)** Glossary • **(IAF 1200)** Assurance Levels **NORMATIVE**: • **(IAF 1300)** Assurance Assessment Scheme • **(IAF 1400)** Assessor Qualifications & Requirements • **(IAF 1500)** Service Assessment Criteria

APPROVED APRIL 2010
http://kantarainitiative.org/confluence/x/e4R7Ag

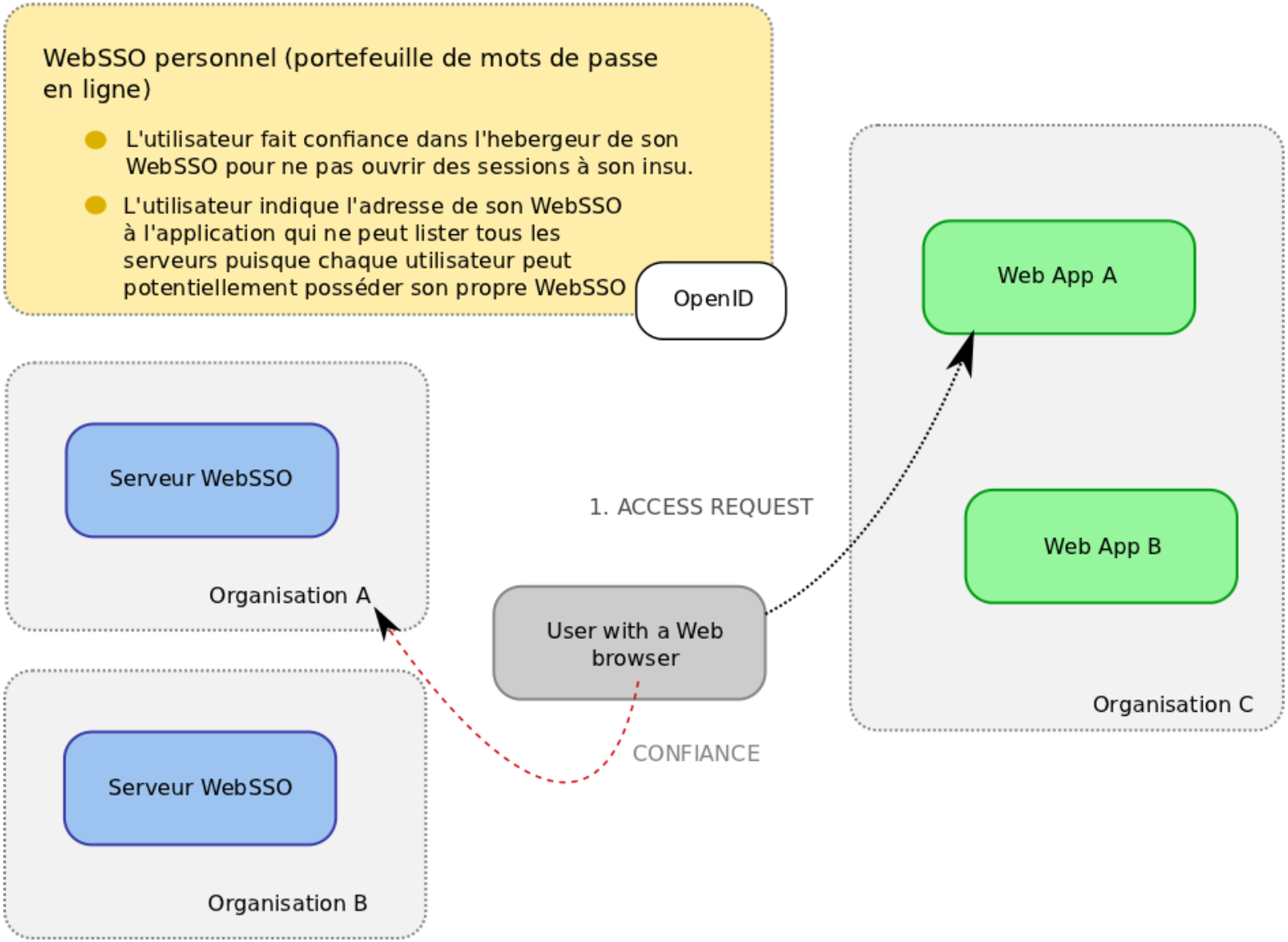
kantara INITIATIVE

Sommaire

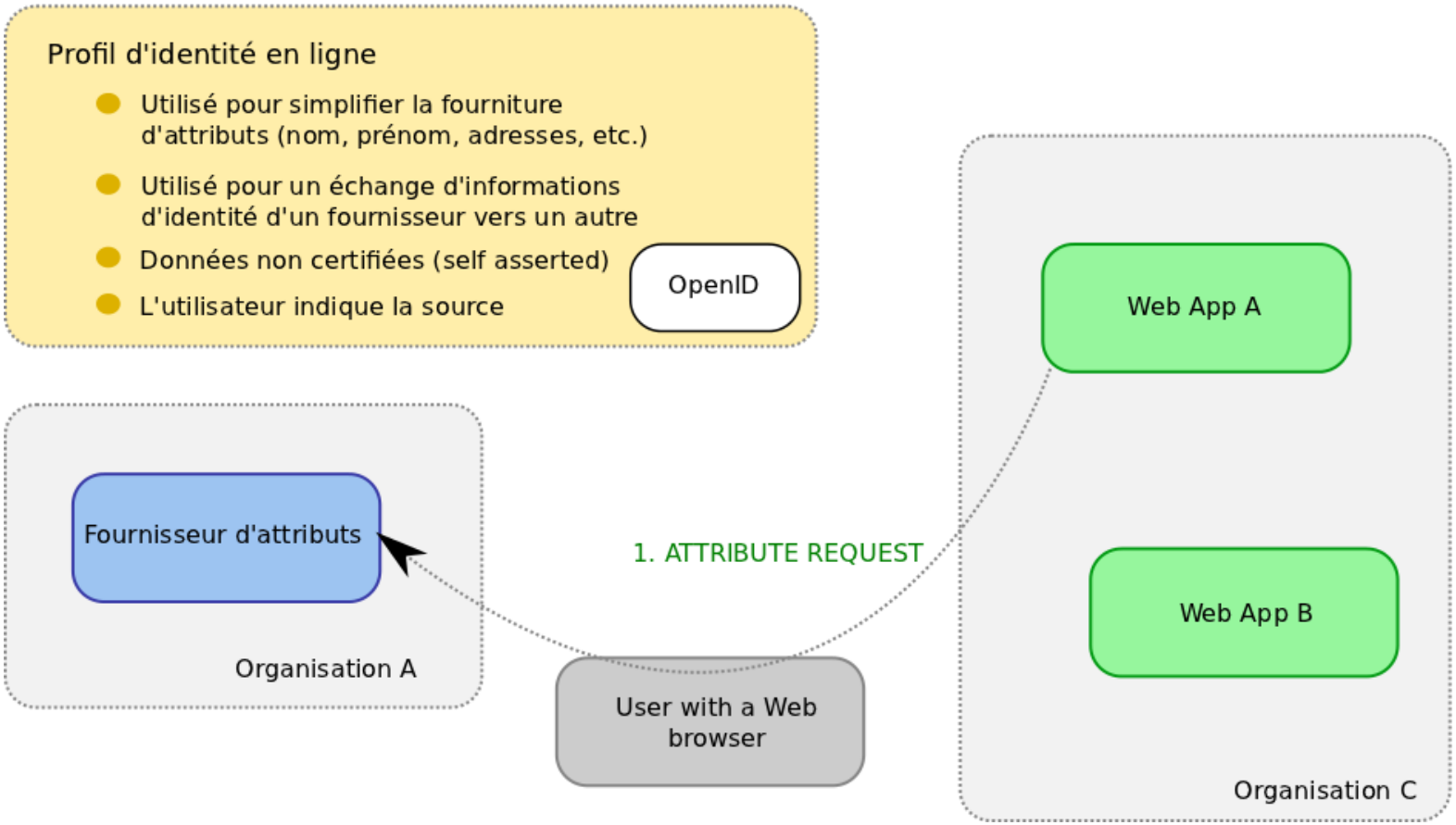
SSO et délégation de l'authentification



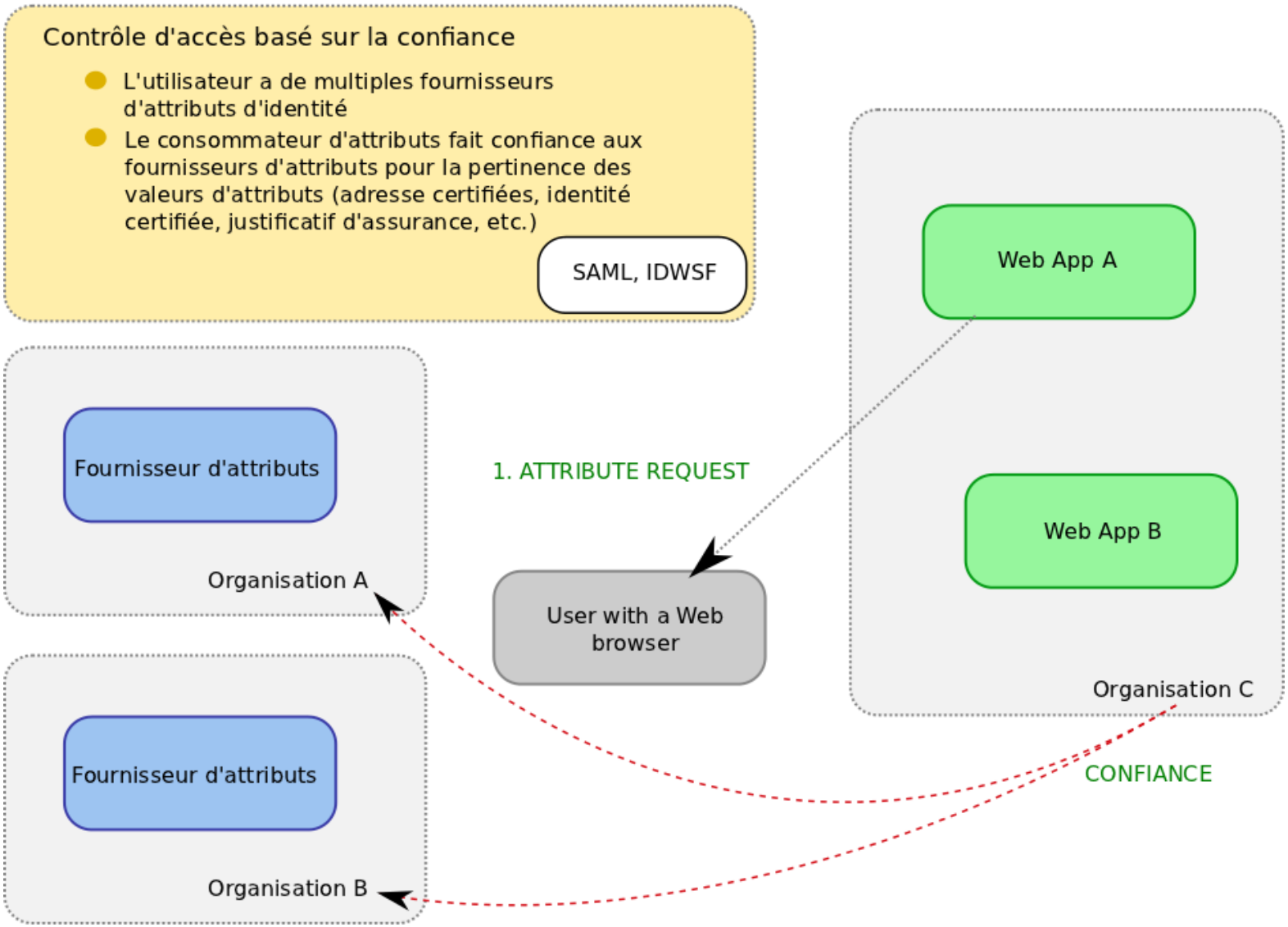
SSO et délégation de l'authentification



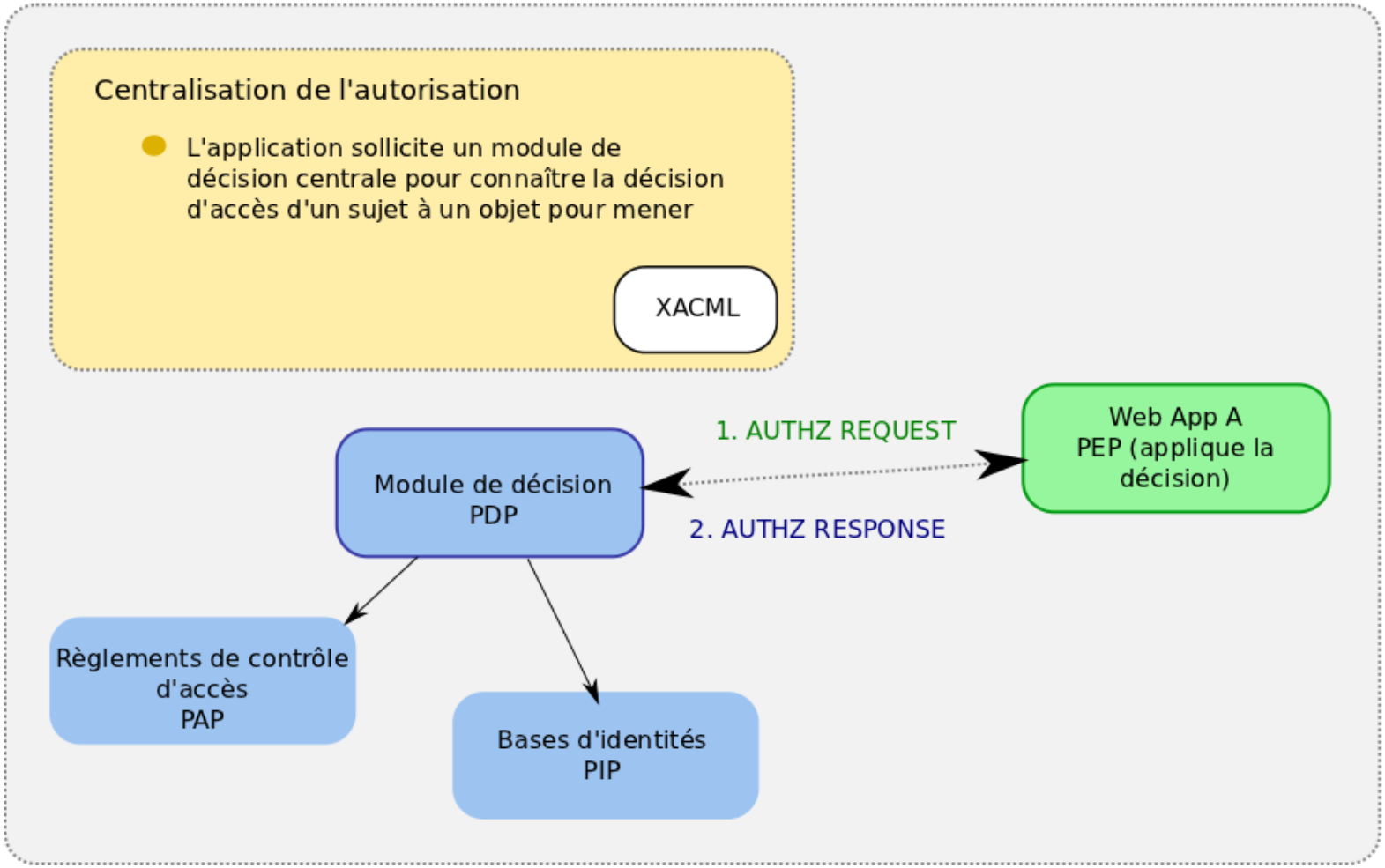
Partage d'attributs d'identité



Partage d'attributs d'identité

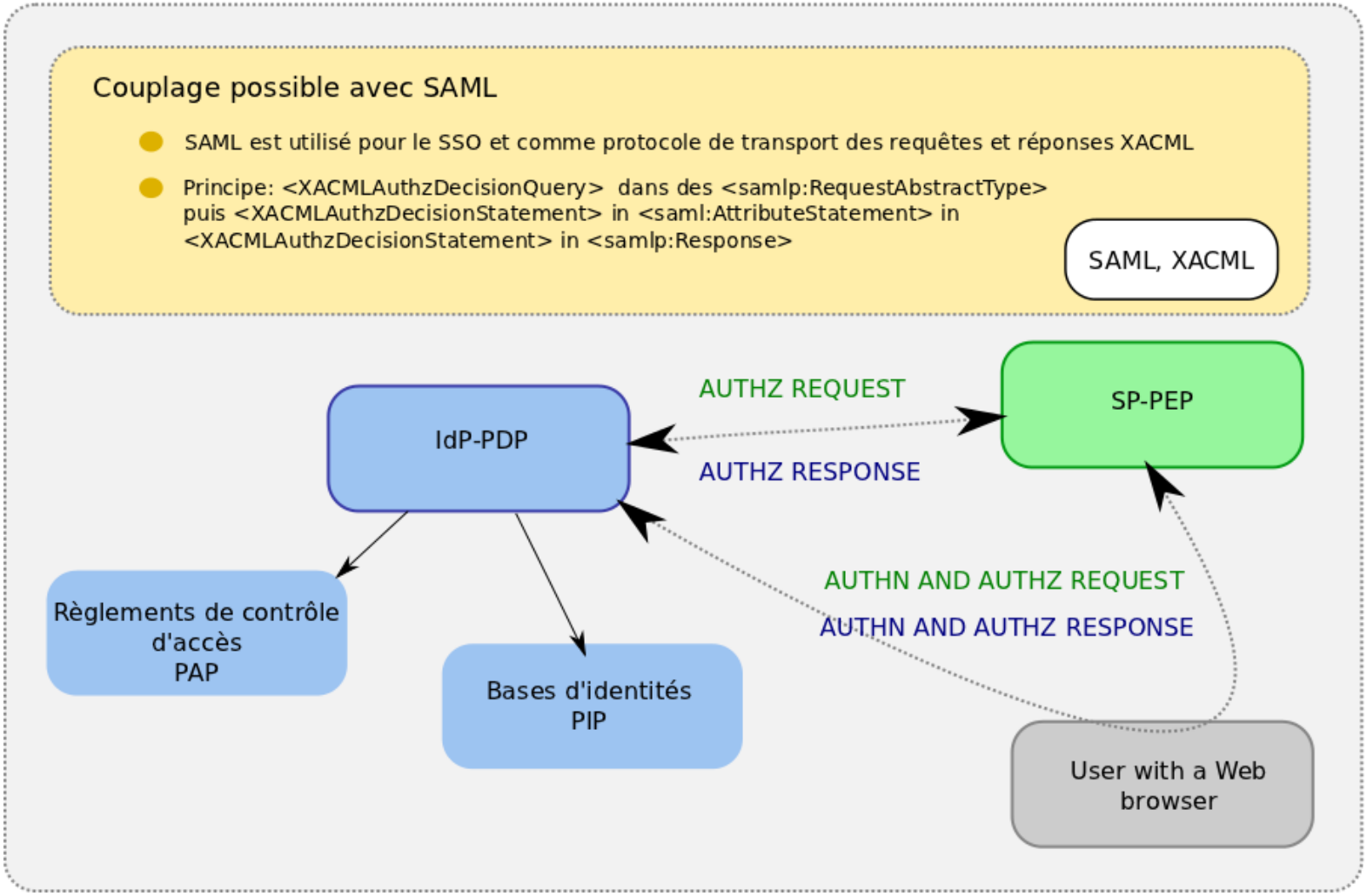


Contrôle d'accès centralisé Web



Ates Mikaël, Dauvergne Benjamin

Contrôle d'accès centralisé Web



Contrôle d'accès centralisé Web

```
pdp_metadata = '''
<?xml version="1.0"?>

<EntityDescriptor xmlns="urn:oasis:names:tc:SAML:2.0:metadata"
    xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    entityID="http://idp5/metadata">

<PDPDescriptor
    protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">

<KeyDescriptor use="signing">...  </KeyDescriptor>
<KeyDescriptor use="encryption">...</KeyDescriptor>

<AuthzService Binding="urn:oasis:names:tc:SAML:2.0:bindings:SOAP"
  Location="http://idp6/authzService"/>

<AssertionIDRequestService Binding="urn:oasis:names:tc:SAML:2.0:bindings:URI"
  Location="http://idp6/PDPAuthAssertionIDRequestService"/>

<NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:persistent</NameIDFormat>

</PDPDescriptor>'''
```

Contrôle d'accès centralisé Web

```
request = saml_authorization_query(sp_metadata, private_key, pdp_metadata,
  name_id='#A1256F', action='access', resource='http://sp.lan/data/1')
request_msg, url = next(request)

def saml_authorization_query_async(sp_metadata_file, private_key,
        pdp_metadata_file, name_id, action, resource, action_ns=None):
    server = lasso.Server.newFromBuffers(sp_metadata_file, private_key)
    server.addProviderFromBuffer(lasso.PROVIDER_ROLE_AUTHZ_AUTHORITY,
            metadata_file)
    assertion_query = lasso.AssertionQuery(server)
    assertion_query.initRequest(None, lasso.HTTP_METHOD_SOAP,
            lasso.ASSERTION_QUERY_REQUEST_TYPE_AUTHZ_DECISION)
    assertion_query.request.subject = lasso.Saml2Subject()
    assertion_query.request.subject.nameId = name_id
    assertion_query.request.action = lasso.Saml2Action.newWithString(action)
    assertion_query.request.action.namespace = action_ns
    assertion_query.request.resource = resource
    assertion_query.buildRequestMsg()
    soap_response = yield assertion_query.msgBody, assertion_query.msgUrl
    assertion_query.processResponseMsg(soap_response)
    assertion = assertion_query.response.assertion[0]
    authz_decision_statement = assertion.authzDecisionStatement[0]
    yield authz_decision_statement.decision
```

Contrôle d'accès centralisé Web

```
pdp = SamlAuthorizationQueryProcessor(pdp_metadata, private_key, sp_metadata)
pdp.process(request_msg, force_decision=INDETERMINATE)

class SamlAuthorizationQueryProcessor:
    def __init__(self, pdp_metadata_file, private_key, sp_metadata_file):
        server = lasso.Server.newFromBuffers(pdp_metadata_file, private_key)
        server.addProviderFromBuffer(lasso.PROVIDER_ROLE_SP, sp_metadata_file)
        self.server = server

    def process(self, soap_message, force_decision=None):
        assertion_query = lasso.AssertionQuery(self.server)
        assertion_query.processRequestMsg(soap_message)
        assertion_query.validateRequest()
        name_id = assertion_query.request.subject.nameId
        action = assertion_query.request.action.content
        action_ns = assertion_query.request.action.namespace or \
                lasso.SAML2_ACTION_NAMESPACE_RWEDC_NEGATION
        resource = assertion_query.request.resource
        if force_decision:
            decision = force_decision
        else:
            decision = self.decide(name_id, action, action_ns, resource)
        assert decision in PERMISSIONS
        authz_decision_statement = lasso.Saml2AuthzDecisionStatement()
        authz_decision_statement.action = assertion_query.request.action
        authz_decision_statement.resource = resource
        authz_decision_statement.decision = decision
        assertion = lasso.Saml2Assertion()
        assertion.subject = assertion_query.request.subject
        assertion.authzDecisionStatement = [ authz_decision_statement ]
        assertion_query.response.assertion = [ assertion ]
        assertion_query.buildResponseMsg()
        return assertion_query.msgBody

    def decide(self, name_id, action, action_ns, resource):
        #overload mino!
        return DENY
```

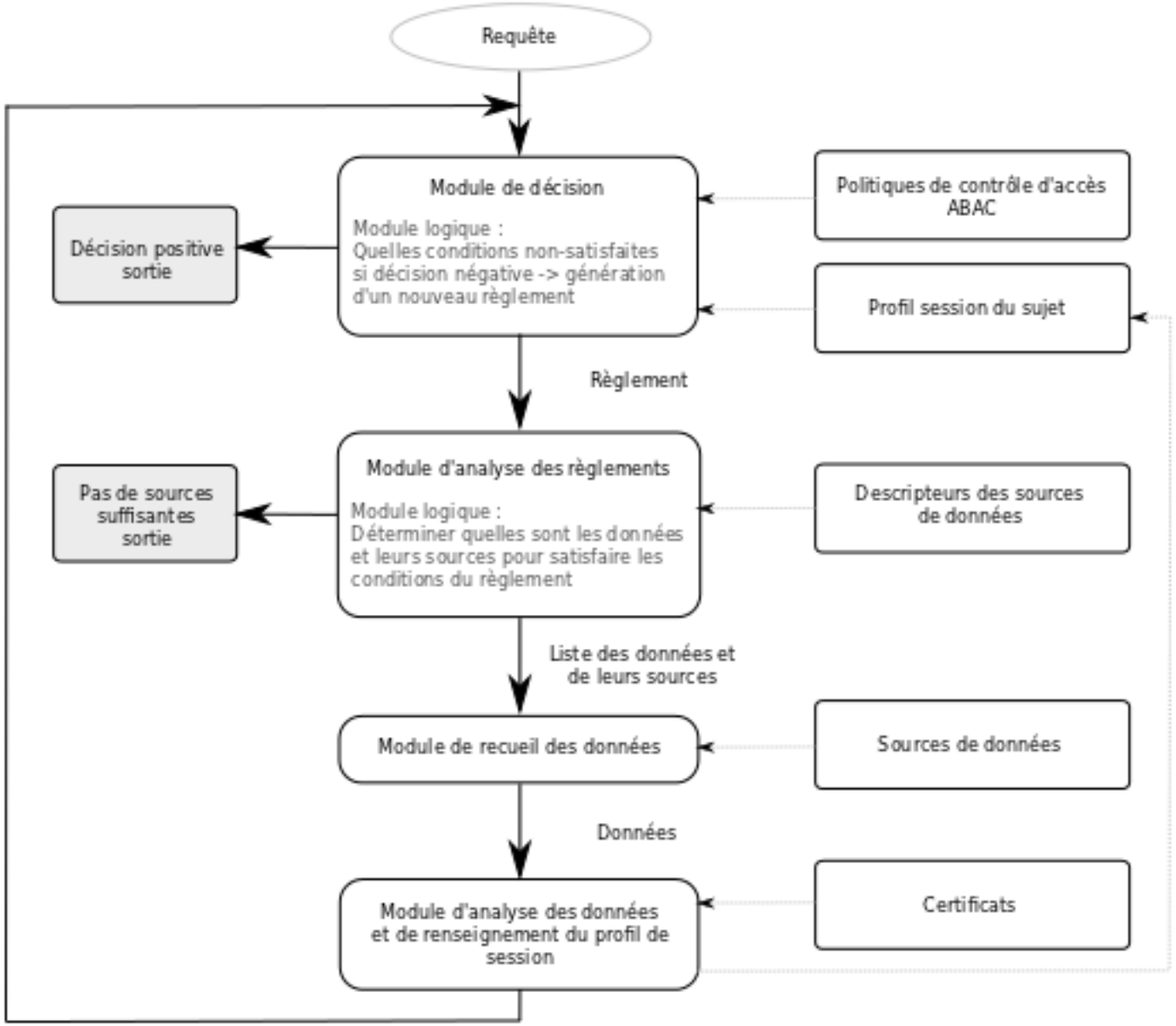
Sommaire

Contrôle d'accès sur les attributs d'identité : ABAC

- Règles pour service d'enregistrement en ligne pour une location de voiture (langage CARL)

```
own r::CNIE issued-by FRgov
own p::DrivingLicence issued-by (Pref42)
own j::HomeJustif issued-by (Elec+, TelcoFilaire+)
own a::AttestRespCivil issued-by (AssocAssurFr)
own m::Money issued-by (VISA, Paypal)
reveal r.pseudonyme, m.amount
where
r.dateOfBirth <= dateMinusYears(today(), 21) AND
isRevocable(r.pseudonyme, FRgov, cdts) AND r.expDate > today() AND
p.expDate > today() AND p.points > 6 AND
j.issueDate > dateMinusDays(today(), 60) AND
isRevocable(j.address, AssocCyberNotaire+, cdts) AND
a.expDate > today() AND
m.amount = amountMoney(transaction) AND
(r.surname, r.firstname) = (p.surname, p.firstname) AND
(r.surname, r.firstname) = (j.surname, j.firstname) AND
(r.surname, r.firstname) = (a.surname, a.firstname)
```

Contrôle d'accès sur les attributs d'identité : ABAC



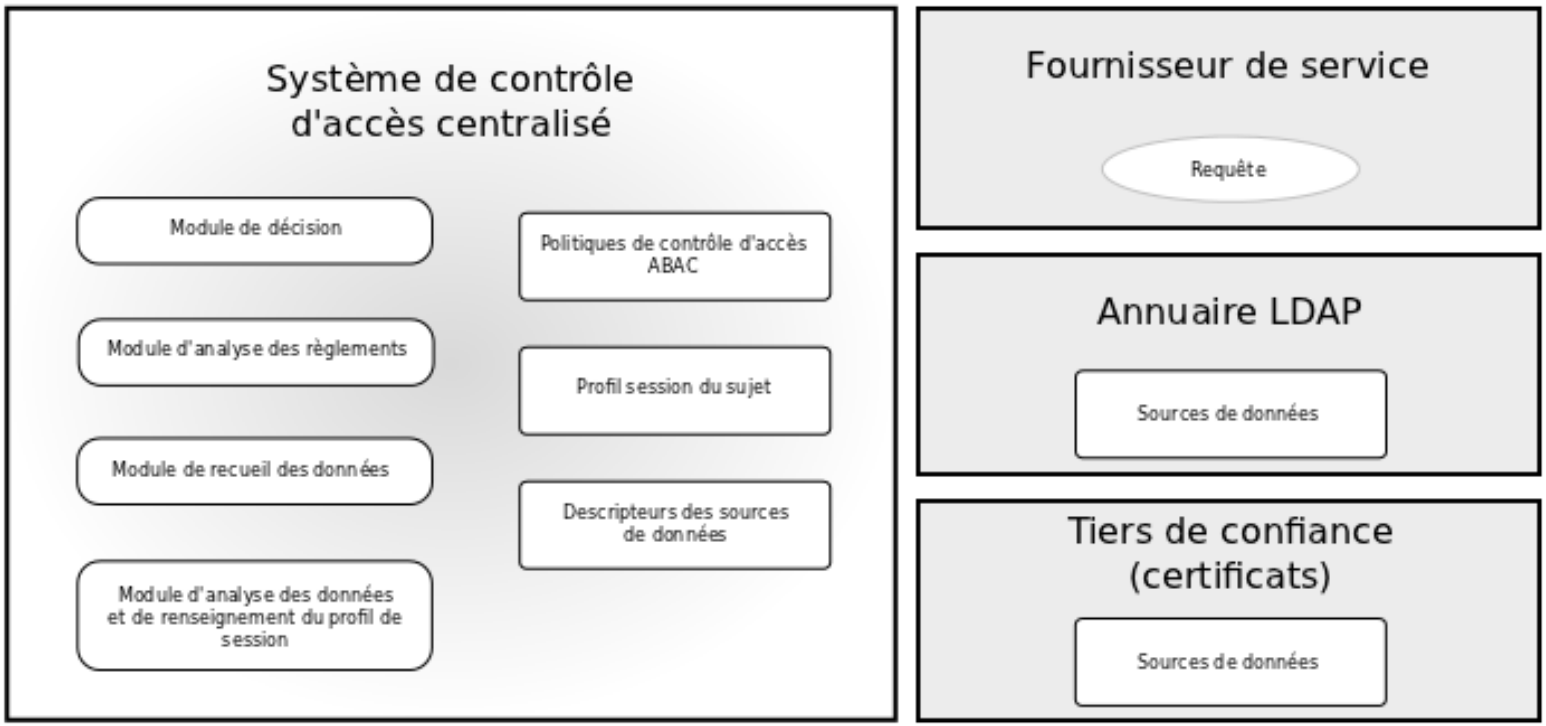
Contrôle d'accès sur les attributs d'identité : ABAC

```
''' (age > 18 of IdP1 ou firtname of IdP2) et (surname of IdP2 ou firtname of IdP2)) '''

s1 = Source(name="IdP1")
s1.save()
s2 = Source(name="IdP2")
s2.save()

def_age = AttributeDefinition(attribute_name='age', source=s1,
    attribute_type=ACS_XACML_DATATYPE_INTEGER)
def_age.save()

def_sn = AttributeDefinition(attribute_name='surname', source=s2,
    attribute_type=ACS_XACML_DATATYPE_STRING)
def_sn.save()

def_fn = AttributeDefinition(attribute_name='firstname', source=s2,
    attribute_type=ACS_XACML_DATATYPE_STRING)
def_fn.save()

rule = Rule()
rule.save()

p_sn = PredicateAny(definition=def_sn, rule=rule)
p_sn.save()

p_fn = PredicateAny(definition=def_fn, rule=rule)
p_fn.save()

age_data = AttributeDataAge(definition=def_age, value='18')
age_data.save()

p_age = PredicateIntegerGreaterThan(val_max=def_age, val_min=age_data,
    rule=rule)
p_age.save()

str_rule = "(%s|%s)&(%s|%s)" % (p_sn.id, p_fn.id, p_age.id, p_fn.id)
rule.expression=str_rule
```

Contrôle d'accès sur les attributs d'identité : ABAC

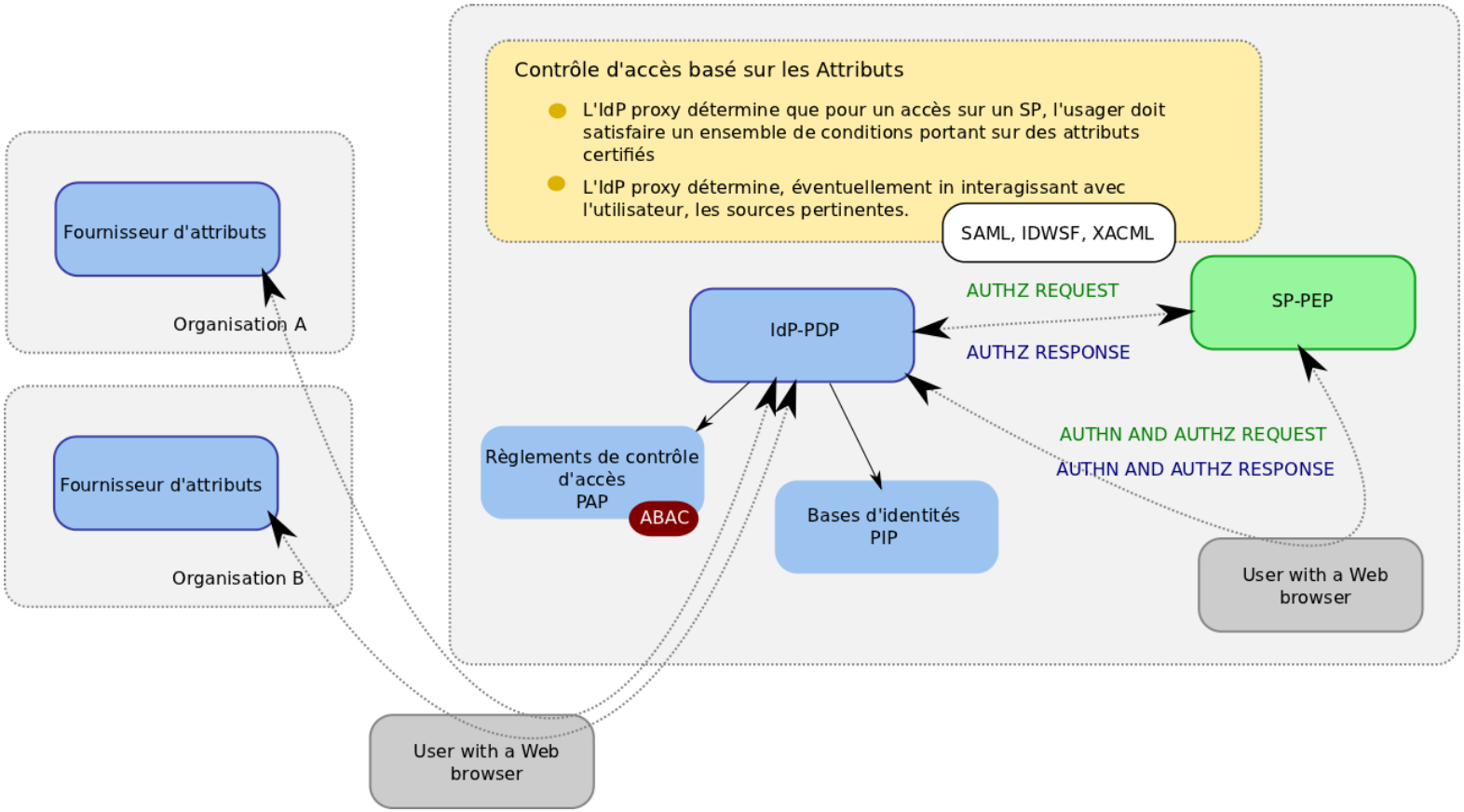
```
profile = SessionProfile()
profile.save()

def_sn2 = AttributeDefinition(attribute_name='surname', source=s2,
    attribute_type=ACS_XACML_DATATYPE_STRING)
def_sn2.save()

sn_data = AttributeDataSurname(definition=def_sn2, value='Ates')
sn_data.save()

def_age2 = AttributeDefinition(attribute_name='age', source=s1,
    attribute_type=ACS_XACML_DATATYPE_INTEGER)
def_age2.save()

age_data2 = AttributeDataAge(definition=def_age2, value='16')
age_data2.save()

profile.data.add(sn_data)
profile.data.add(age_data2)

l = check_predicates(rule, profile)
res = evaluation(rule.expression, l)
print "Authorized? %s" % res
```

ABAC "2-tiers”

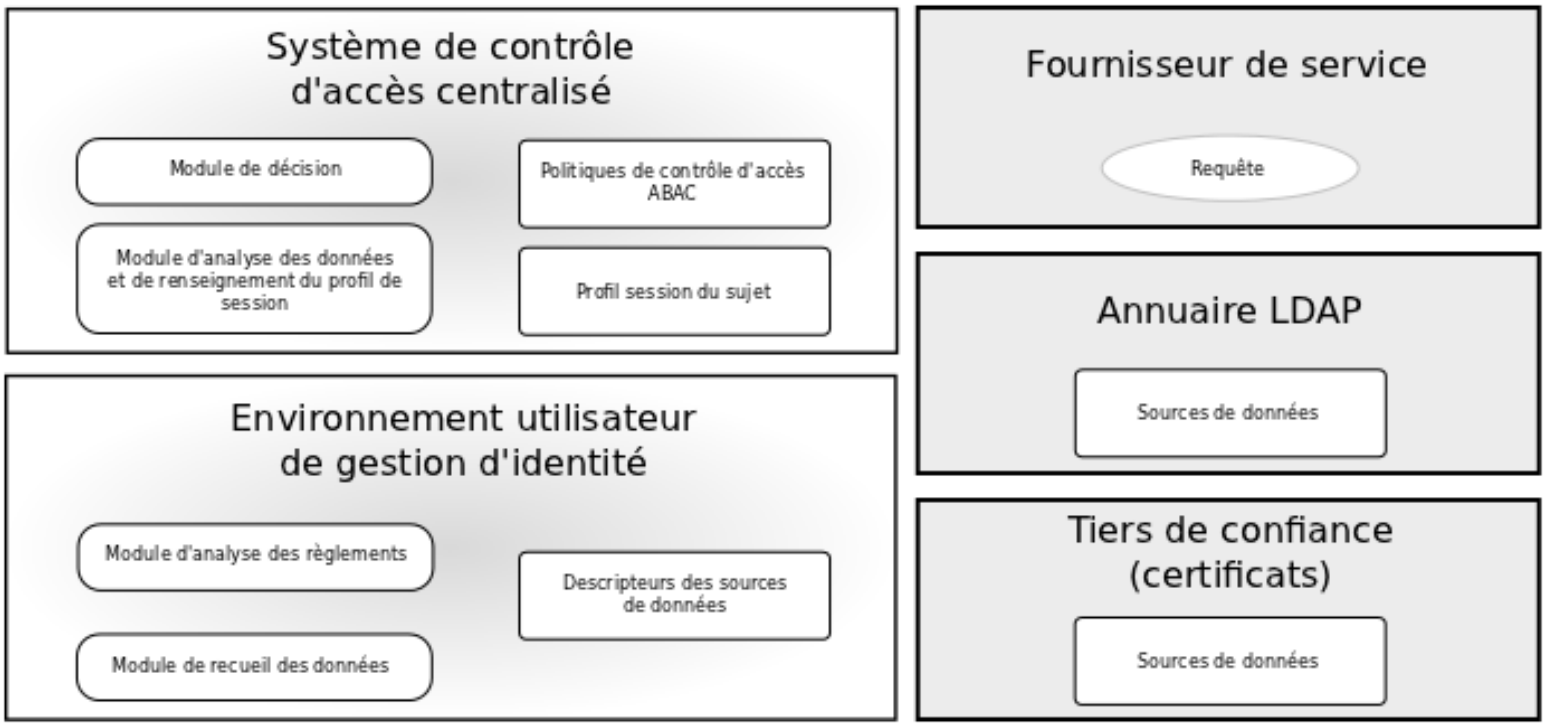


Architecture de décision 2-tiers pour le contrôle d'accès basé sur la confiance.

ABAC "2-tiers”

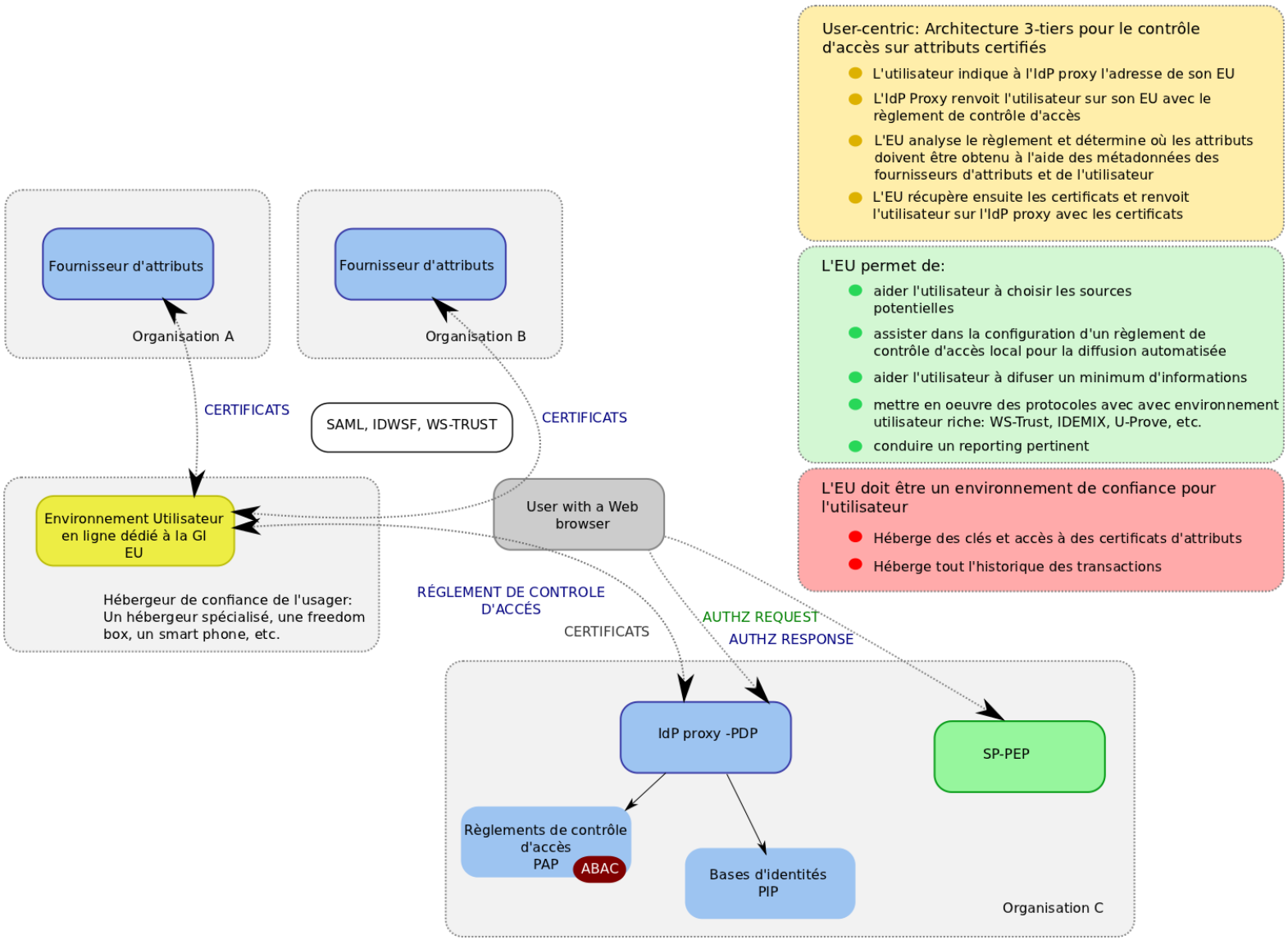

Contrôle d'accès basé sur les Attributs
L'IdP proxy détermine que pour un accès sur un SP, l'usager doit satisfaire un ensemble de conditions portant sur des attributs certifiés
L'IdP proxy détermine, éventuellement in interagissant avec l'utilisateur, les sources pertinentes.
SAML, IDWSF, XACML
Fournisseur d'attributs
Organisation A
Fournisseur d'attributs
Organisation B
IdP-PDP
SP-PEP
AUTHZ REQUEST
AUTHZ RESPONSE
Règlements de contrôle d'accès PAP
ABAC
Bases d'identités PIP
AUTHN AND AUTHZ REQUEST
AUTHN AND AUTHZ RESPONSE
User with a Web browser
User with a Web browser

ABAC "3-tiers”

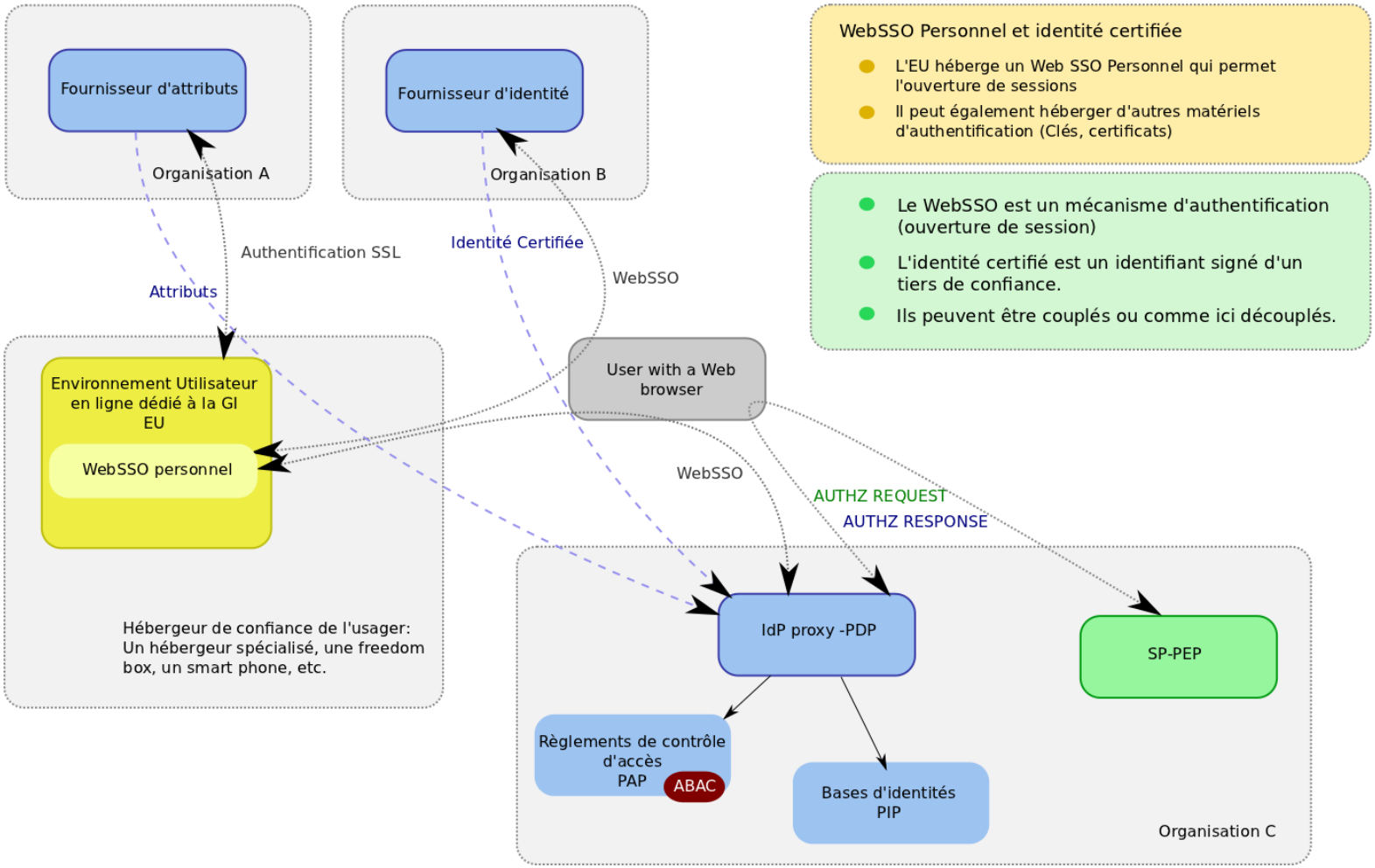


Architecture de décision 3-tiers pour le contrôle d'accès basé sur la confiance.

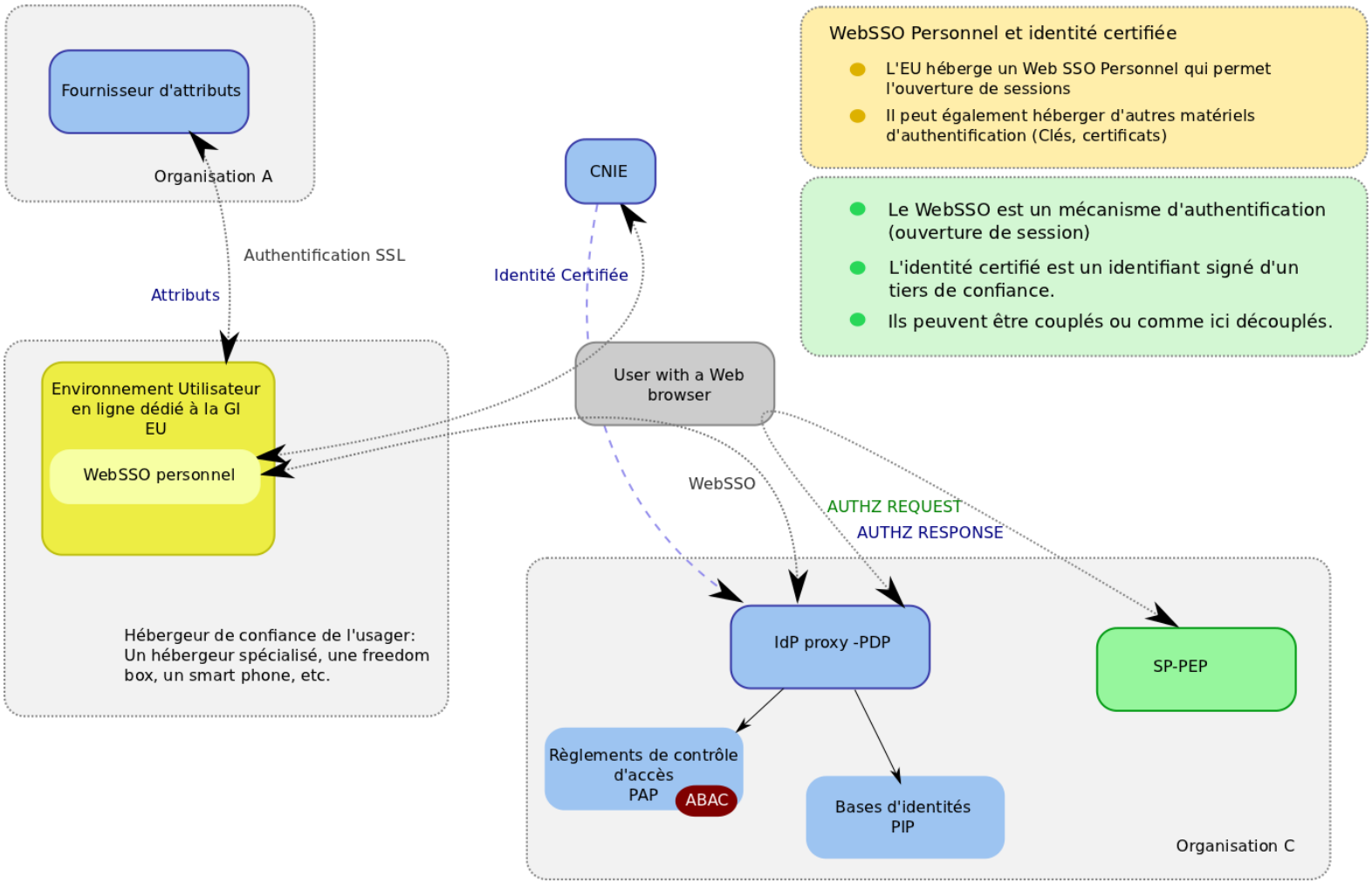
ABAC "3-tiers" : Architecture User Centric



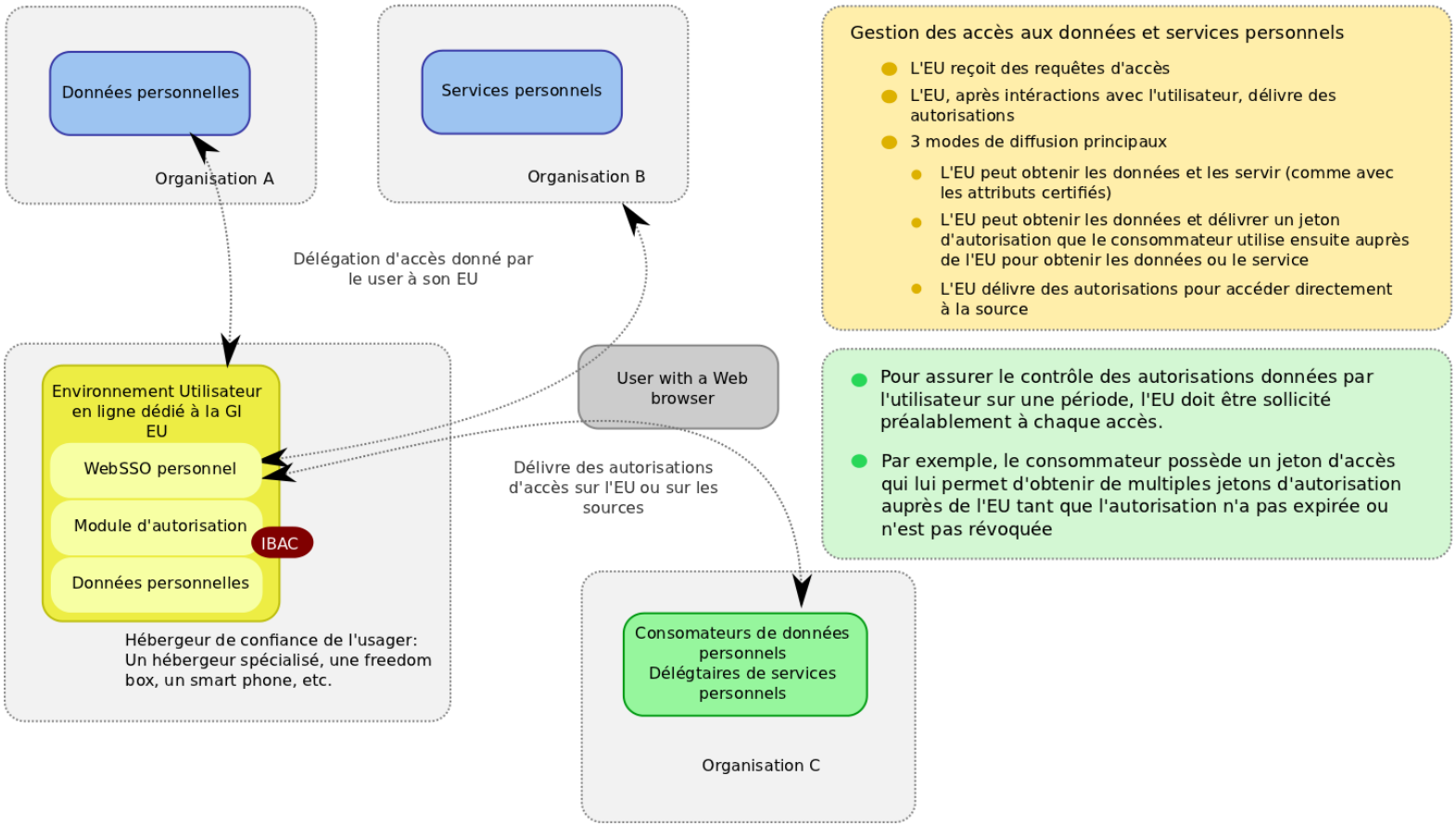
Architecture User Centric



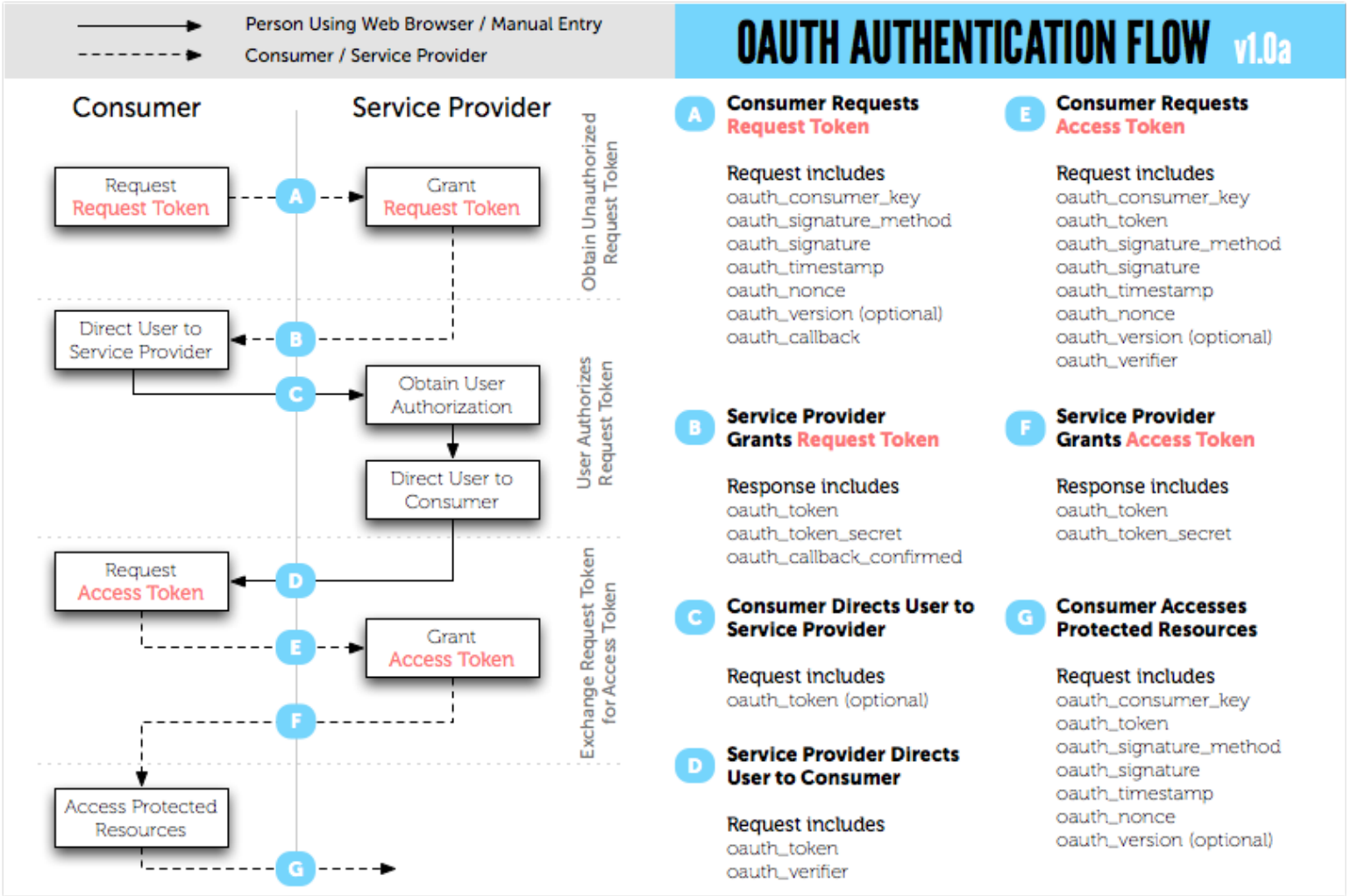
Architecture User Centric



Architecture User Centric

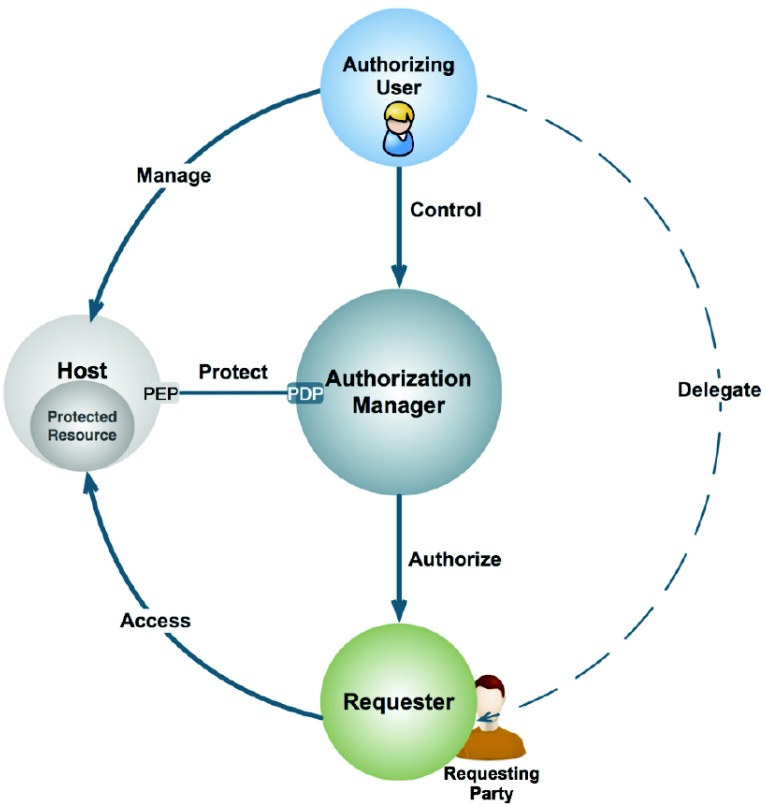


Architecture User Centric

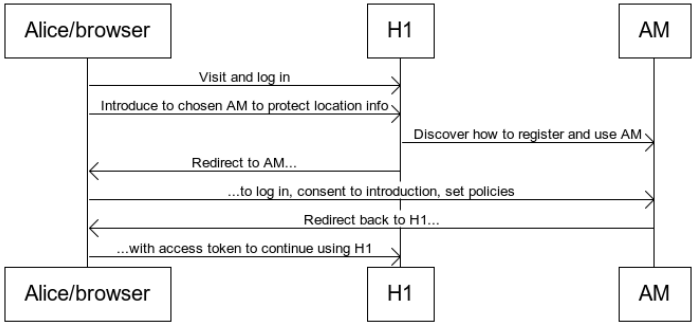


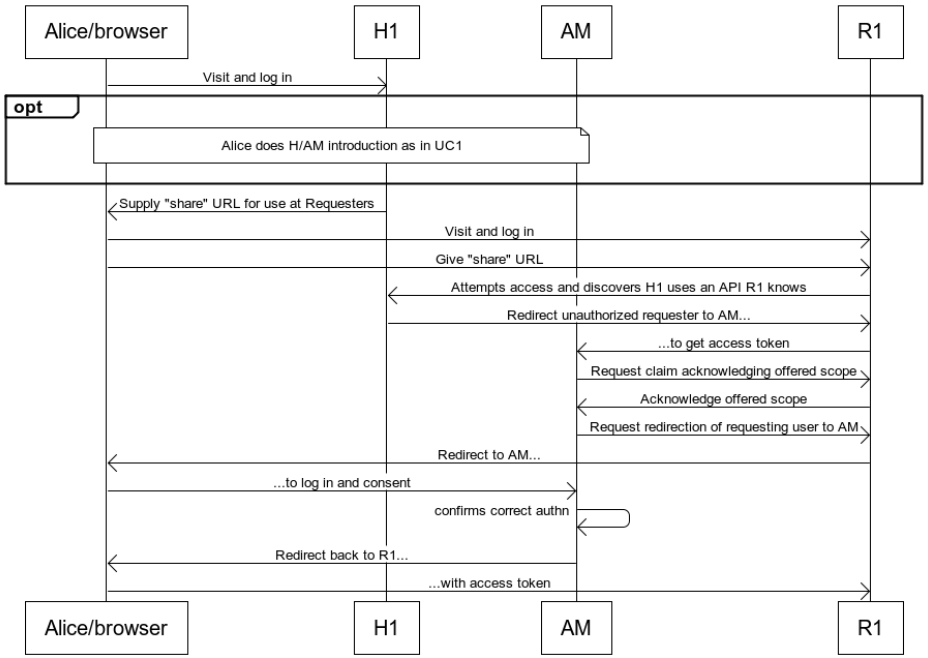
Architecture User Centric

- User Managed Access (UMA) est un WG de Kantara travaillant sur des spécifications éponymes basée sur OAUTH2.0
- Plus riche que du proxying OAUTH



User Managed Access





Sommaire

LASSO

Librairie SAML2 et IDWSF2

- Implémentation en C des spécifications SAML2 (Conformance 2005) et IDWSF2
- Bindings en Python, Perl, Java et PHP
- GNU GPL2

Authentic1

Fournisseur d'identité SAML2 et IDWSF2

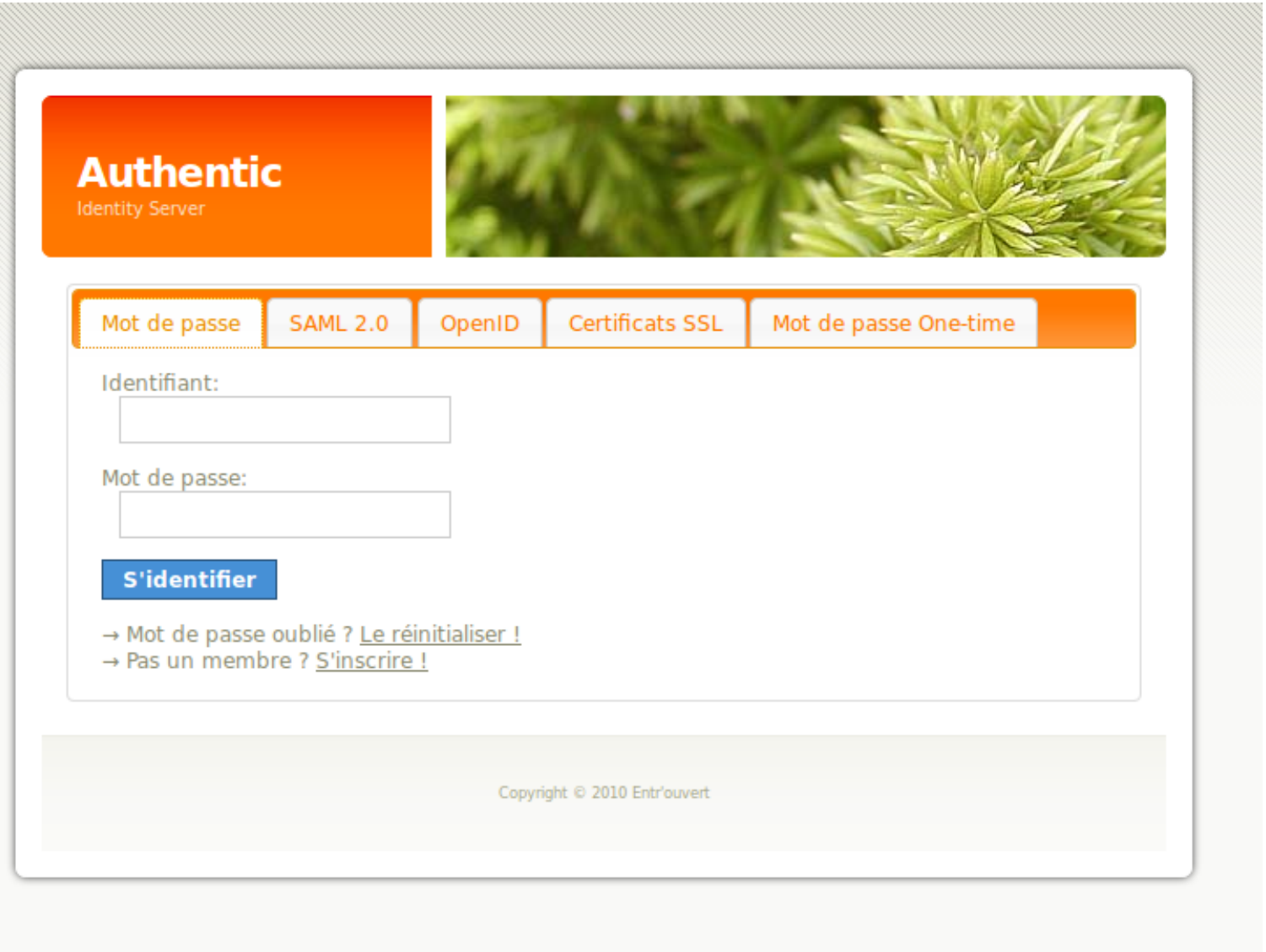
- Implémentation en python sur Quixote d'un IDP SAML2 et DS IDWSF2.
- IdP SAML2 proxy.
- GNU GPL2

Authentic2

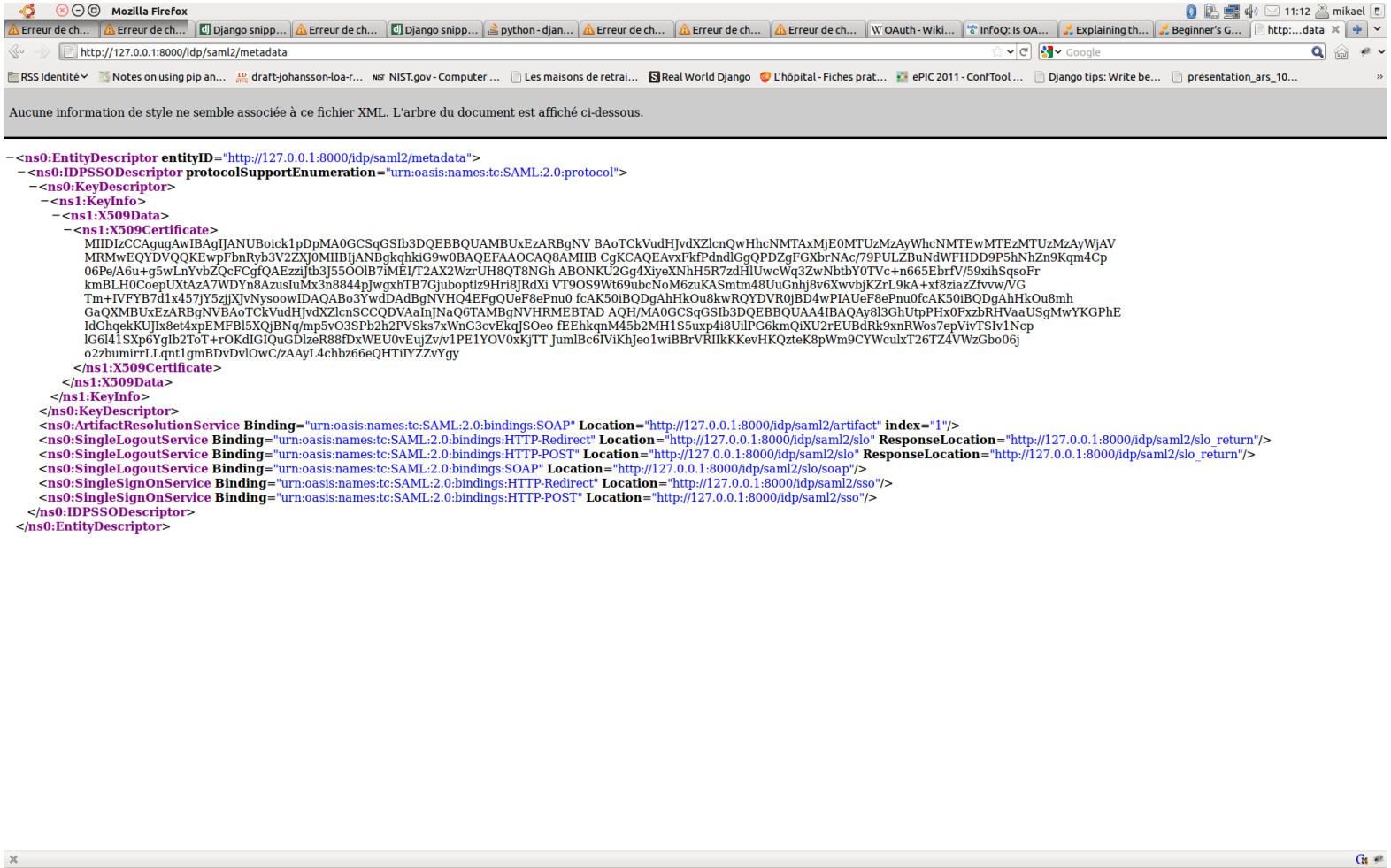
Plateform de gestion d'identité centralisée

- Implémentation en python sur Django
- Objectif : Quelque soit le protocole d'identité, assumer les différents rôles de ce protocole
- Hub de protocoles : peut jouer le rôle de proxy (coupler les deux roles source et destination)
- Aujourd'hui, supporte SAML2, OpenID, CAS, X509, OATH
- GNU AGPL3

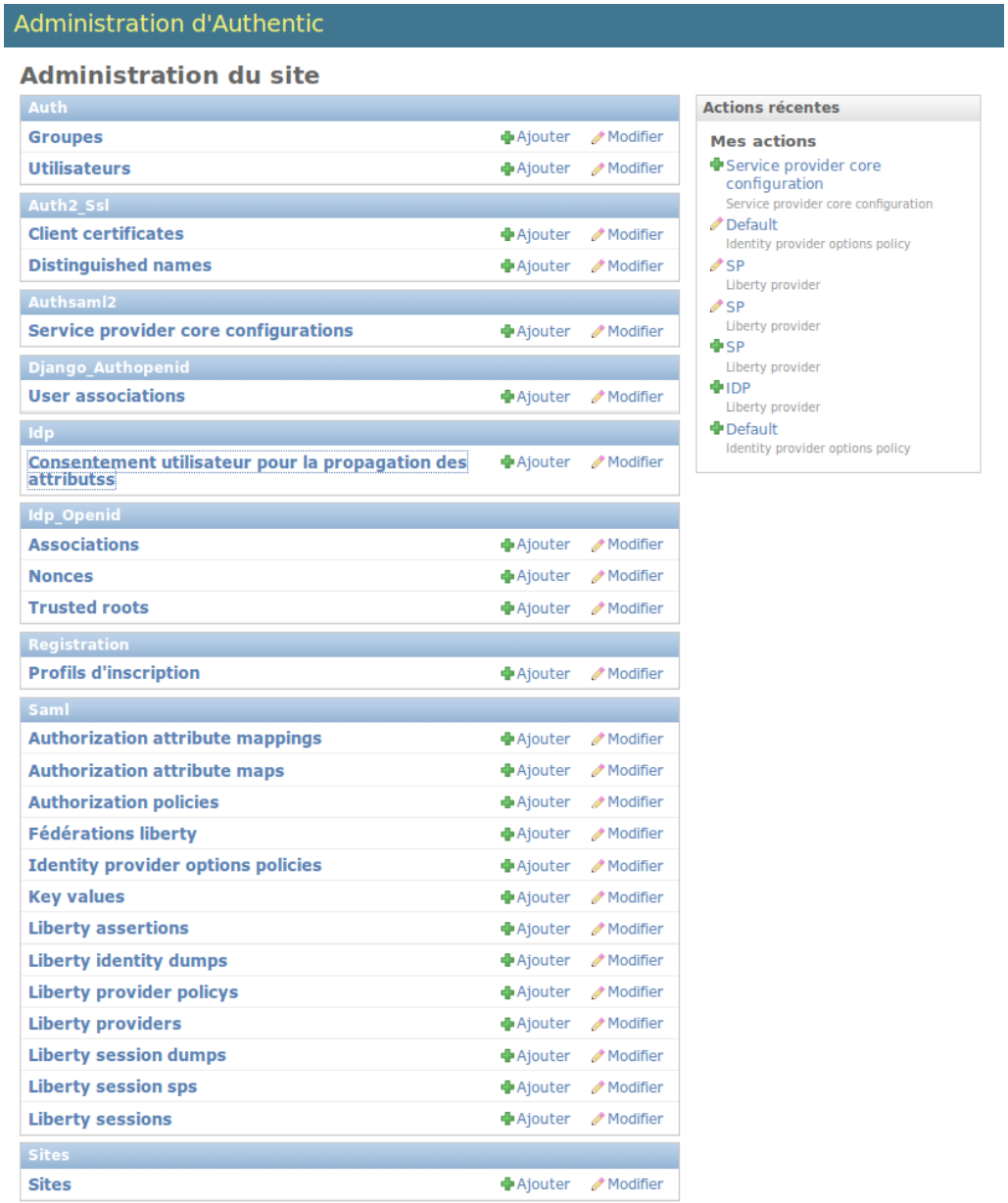
Authentic2



Authentic2

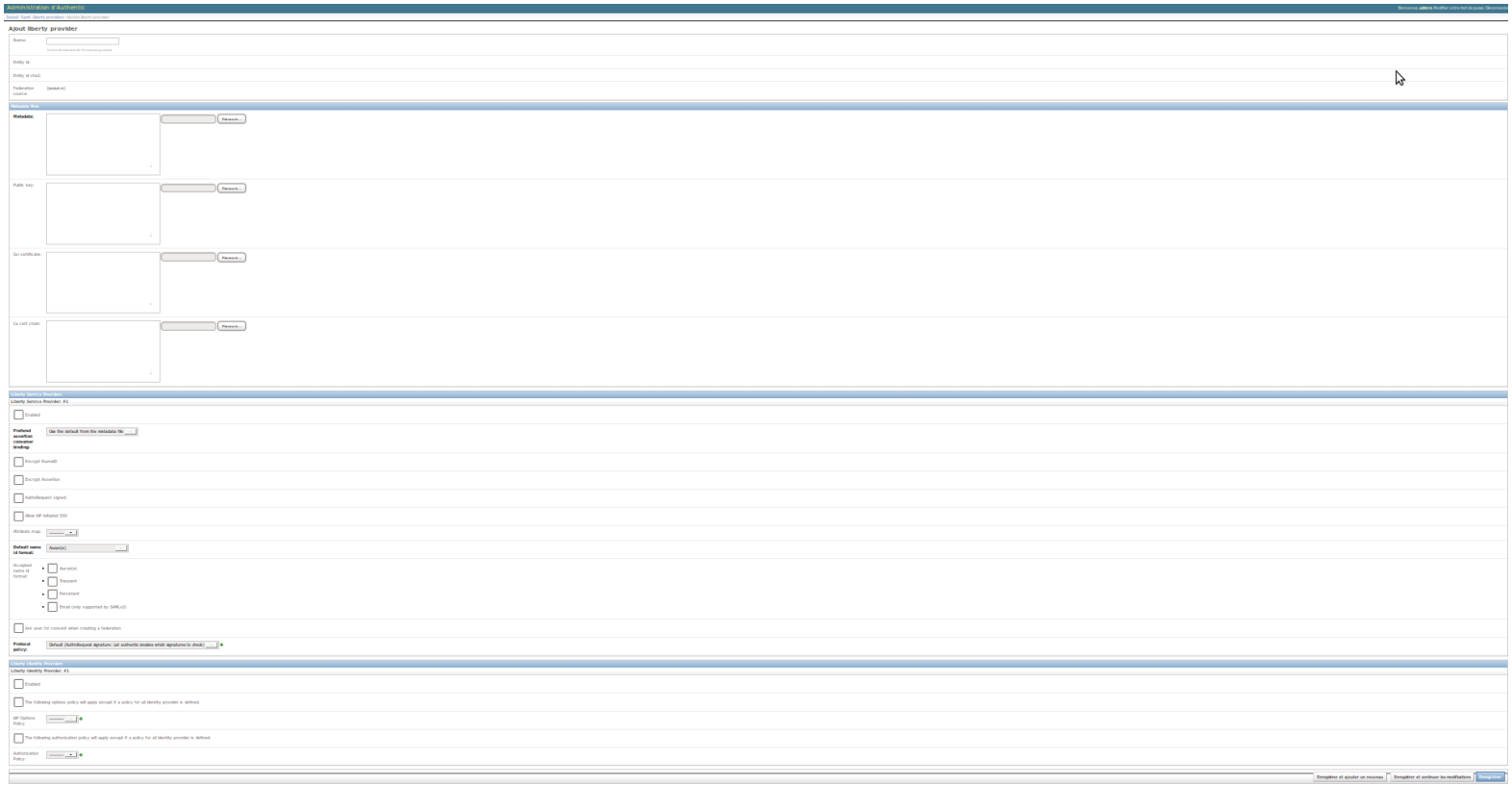

Mozilla Firefox
http://127.0.0.1:8000/idp/saml2/metadata
Aucune information de style ne semble associée à ce fichier XML. L'arbre du document est affiché ci-dessous.
−<ns0:EntityDescriptor entityID="http://127.0.0.1:8000/idp/saml2/metadata">
−<ns0:IDPSSODescriptor protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
−<ns0:KeyDescriptor>
−<ns1:KeyInfo>
−<ns1:X509Data>
−<ns1:X509Certificate>
MIIDIzCCAgugAwIBAgIJANUBoick1pDpMA0GCSqGSIb3DQEBBQUAMBUxEzARBgNV BAoTCkVudHJvdXZlcnQwHhcNMTAxMjE0MTUzMzAyWhcNMTEwMTEzMTUzMzAyWjAV
MRMwEQYDVQQKEwpFbnRyb3V2ZXJ0MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB CgKCAQEAvxFkfPdndlGgQPDZgFGXbrNAc/79PULZBuNdWFHDD9P5hNhZn9Kqm4Cp
06Pe/A6u+g5wLnYvbZQcFCgfQAEzzijtb3J55OOlB7iMEI/T2AX2WzrUH8QT8NGh ABONKU2Gg4XiyeXNhH5R7zdHlUwcWq3ZwNbtbY0TVc+n665EbrfV/59xihSqsoFr
kmBLH0CoepUXtAzA7WDYn8AzusIuMx3n8844pJwgxhTB7Gjuboptlz9Hri8JRdXi VT9OS9Wt69ubcNoM6zuKASmtm48UuGnhj8v6XwvbjKZrL9kA+xf8ziazZfvvw/VG
Tm+IVFYB7d1x457jY5zjjXJvNysoowIDAQABo3YwdDAdBgNVHQ4EFgQUeF8ePnu0 fcAK50iBQDgAhHkOu8kwRQYDVR0jBD4wPIAUeF8ePnu0fcAK50iBQDgAhHkOu8mh
GaQXMBUxEzARBgNVBAoTCkVudHJvdXZlcnSCCQDVAaInJNaQ6TAMBgNVHRMEBTAD AQH/MA0GCSqGSIb3DQEBBQUAA4IBAQAy8l3GhUtpPHx0FxzbRHVaaUSgMwYKGPhE
IdGhqekKUJIx8et4xpEMFBl5XQjBNq/mp5vO3SPb2h2PVSks7xWnG3cvEkqJSOeo fEEhkqnM45b2MH1S5uxp4i8UilPG6kmQiXU2rEUBdRk9xnRWos7epVivTSIv1Ncp
lG6l41SXp6YgIb2ToT+rOKdIGIQuGDlzeR88fDxWEU0vEujZvJv1PE1YOV0xKjTT JumlBc6IViKhJeo1wiBBrVRIlkKKevHKQzteK8pWm9CYWculxT26TZ4VWzGbo06j
o2zbumirrLLqnt1gmBDvDvlOwC/zAAyL4chbz66eQHTiIYZZvYgy

Authentic2



Ates Mikaël, Dauvergne Benjamin

Authentic2



Larpe

Intégration

- Implémentation en python sur Quixote d'un reverse proxy SAML2
- Intégration du SSO aux applications Web existantes
- GNU AGPL3

Authentic2

Intégration

- Utilisation de Lasso en python et intégration avec authentic2 (partie SP).

Intégration

```
def sso(request):
    entity_id = request.REQUEST.get('entity_id')
    server = build_service_provider(request)
    p = load_provider(request, entity_id, server=server, sp_or_idp='idp', autoload=True)
    login = lasso.Login(server)
    http_method = server.getFirstHttpMethod(server.providers[p.entity_id],
                lasso.MD_PROTOCOL_TYPE_SINGLE_SIGN_ON)
    login.initAuthnRequest(p.entity_id, http_method)
    setAuthnrequestOptions(p, login, force_authn, is_passive)
    login.buildAuthnRequestMsg()
    return return_saml2_request(request, login)
```

Intégration

```
def singleSignOnPost(request):
    server = build_service_provider(request)
    login = lasso.Login(server)
    message = get_saml2_post_response(request)
    login.processAuthnResponseMsg(message)
    provider_id = login.remoteProviderId
    provider_loaded = load_provider(request, provider_id, server=server, sp_or_idp='idp', autoload=True)
    s = get_service_provider_settings()

    assertion = login.response.assertion[0]
    '''Checkings'''

    attributes = {}
    for att_statement in login.assertion.attributeStatement:
        for attribute in att_statement.attribute:
            name, format, nickname = \
                    attribute.name.decode('ascii'), \
                    attribute.nameFormat.decode('ascii'), \
                    attribute.friendlyName
            values = attribute.attributeValue
            if values:
                attributes[(name, format)] = []
                if nickname:
                    attributes[nickname] = attributes[(name, format)]
            for value in values:
                content = []
                for any in value.any:
                    content.append(any.exportToXml())
               content = ''.join(content)
               attributes[(name, format)].append(content.decode('utf8'))
    request.session['attributes'] = attributes

    decisions = signals.authz_decision.send(sender=None,
         request=request, attributes=attributes, provider=provider)
```

Intégration

```
access_granted = True
for decision in decisions:
    dic = decision[1]
    if not dic['authz']:
        access_granted = False
if not access_granted:
    return error_page(request,
        logger=logger, default_message=False, timer=True)

user = request.user
login.acceptSso()
if not user and AuthSAML2PersistentBackend(). \
        create_user(name_id=login.nameIdentifier,
            provider_id=provider.entity_id):
    user = AuthSAML2PersistentBackend(). \
        authenticate(name_id=login.nameIdentifier,
            provider_id=login.remoteProviderId)
auth_login(request, user)
signals.auth_login.send(sender=None, request=request, attributes=attributes)
save_session(request, login, kind=LIBERTY_SESSION_DUMP_KIND_SP)
return HttpResponseRedirect(url)
```

Intégration : Exemple du portail captif univnautes (EDUSPOT)

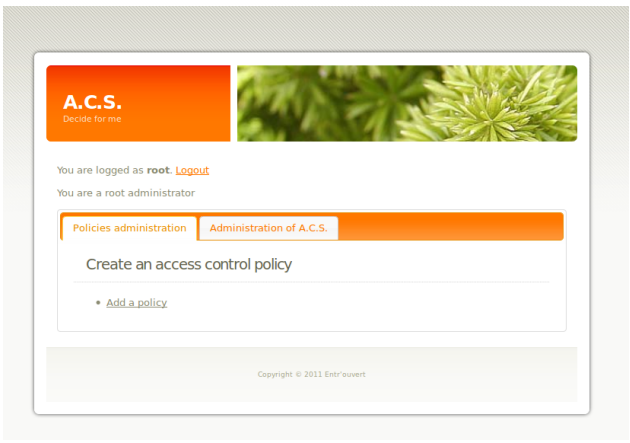
```
def user_login_cb(sender, request, attributes={}, **kwargs):
    eduPersonTargetedID = attributes['__nameid']
    eduPersonTargetedID_NameQualifier = attributes['__issuer']

    user = eduPersonTargetedID + '|' + attributes['__nameid'] + '|' +\
      eduPersonTargetedID_NameQualifier
    ip = request.META['REMOTE_ADDR']

    # open the firewall for this client
    cmd = ['cp_allow', 'ip=%s' % ip, 'username=%s' % user]
    p = subprocess.Popen(cmd, close_fds=True,
                         stdin=subprocess.PIPE,
                         stdout=subprocess.PIPE,
                         stderr=subprocess.PIPE)
    stdout, stderr = p.communicate()
    request.session['pfsenseid'] = stdout
    request.session['prefered_idp'] = attributes['__issuer']
    if 'displayName' in attributes:
        request.session['display_name'] = attributes['displayName'][0]
    return True

signals.auth_login.connect(user_login_cb, dispatch_uid='authentic2.idp')
```

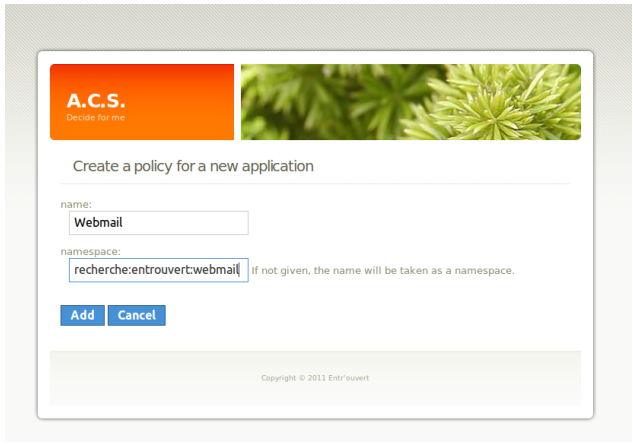
Veridic

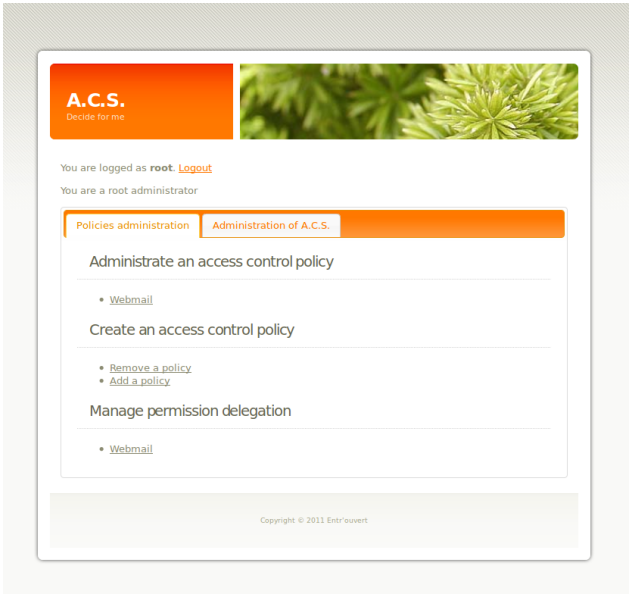
Contrôle d'accès centralisé

- Module de contrôle d'accès centralisé (Django/python)
- Système de définition de politique de contrôle d'accès RBAC et ABAC et de prise de décision
- Auto-administré
- En cours de test
- GNU GPL3

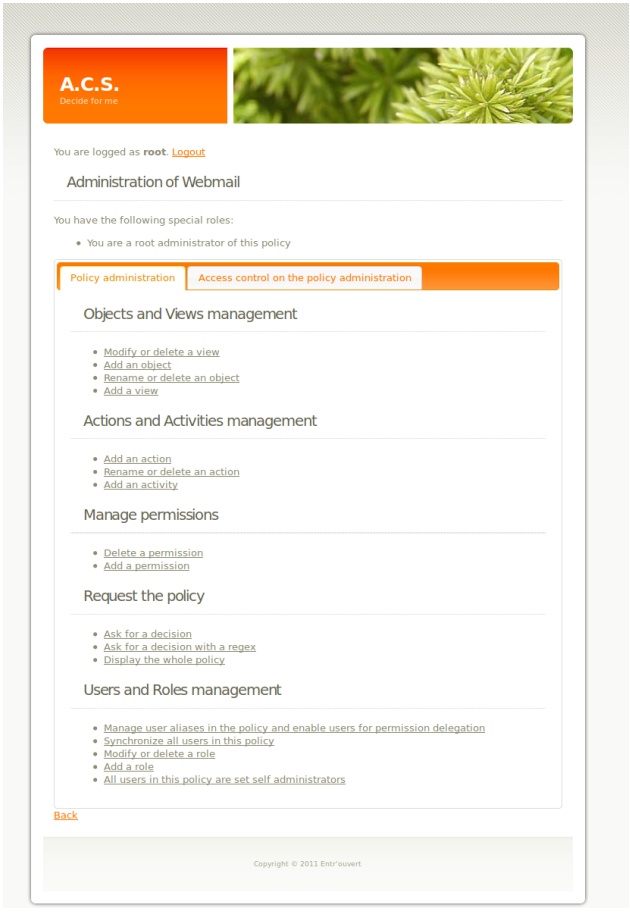
Veridic

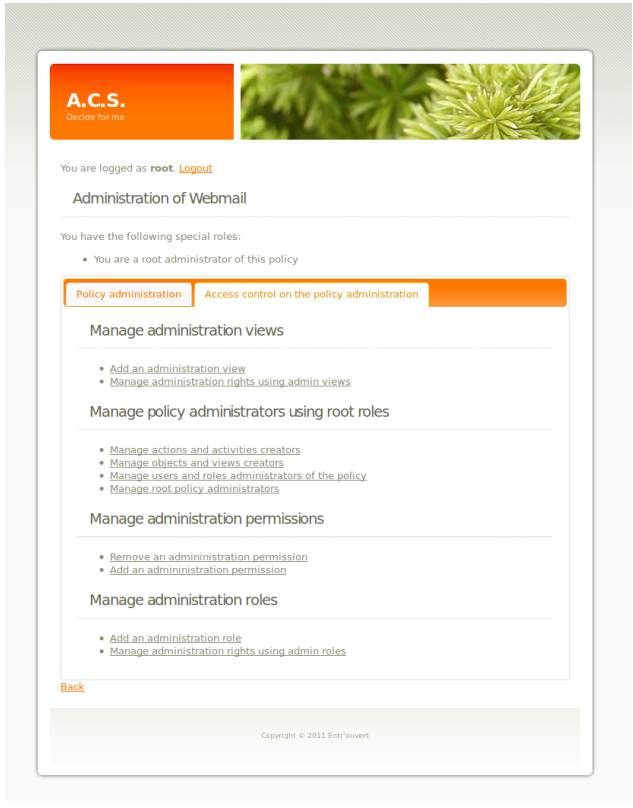

A.C.S.
Decide for me
You are logged as root. Logout
You are a root administrator
Policies administration
Administration of A.C.S.
Create an access control policy
Add a policy
Copyright © 2011 Entr'ouvert


A.C.S.
Decide for me
Create a policy for a new application
name:
Webmail
namespace:
recherche:entrouvert:webmail
If not given, the name will be taken as a namespace.
Add
Cancel
Copyright © 2011 Entr'ouvert

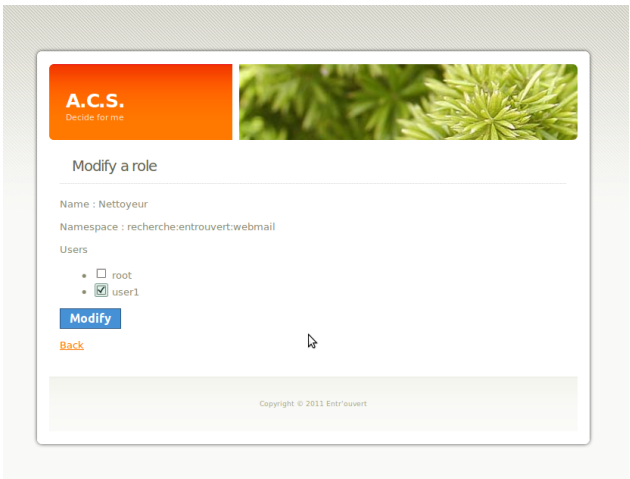

A.C.S.
Decide for me
You are logged as root. Logout
You are a root administrator
Policies administration
Administration of A.C.S.
Administrate an access control policy
Webmail
Create an access control policy
Remove a policy
Add a policy
Manage permission delegation
Webmail
Copyright © 2011 Entr'ouvert

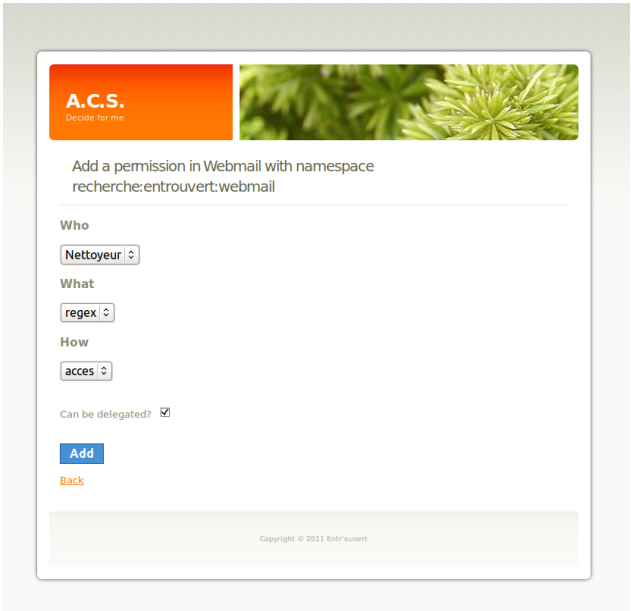
Veridic


A.C.S.
Decide for me
You are logged as root. Logout
Administration of Webmail
You have the following special roles:
You are a root administrator of this policy
Policy administration
Access control on the policy administration
Objects and Views management
Modify or delete a view
Add an object
Rename or delete an object
Add a view
Actions and Activities management
Add an action
Rename or delete an action
Add an activity
Manage permissions
Delete a permission
Add a permission
Request the policy
Ask for a decision
Ask for a decision with a regex
Display the whole policy
Users and Roles management
Manage user aliases in the policy and enable users for permission delegation
Synchronize all users in this policy
Modify or delete a role
Add a role
All users in this policy are set self administrators
Back
Copyright © 2011 Entr'ouvert

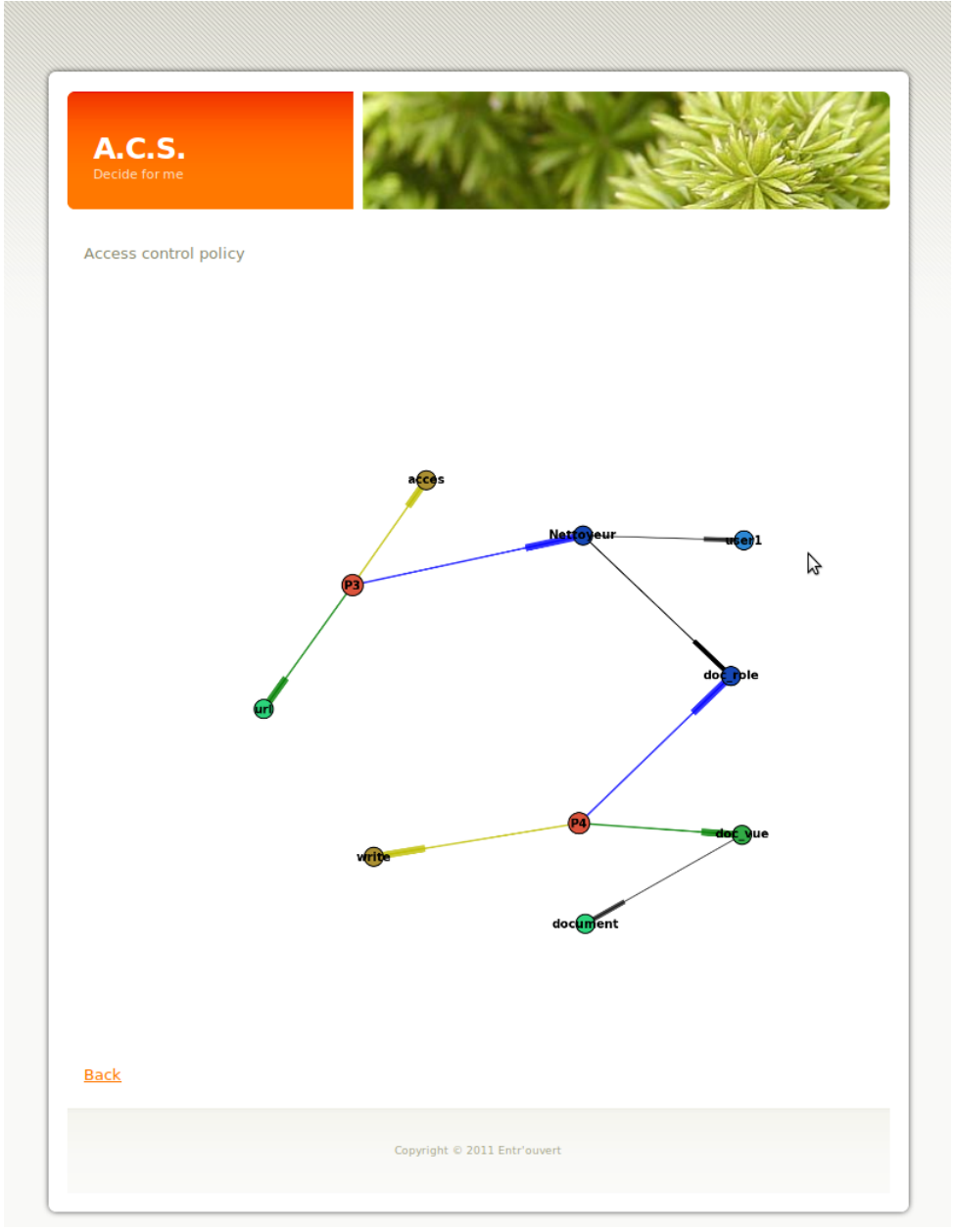

A.C.S.
Decide for me
You are logged as root. Logout
Administration of Webmail
You have the following special roles:
You are a root administrator of this policy
Policy administration
Access control on the policy administration
Manage administration views
Add an administration view
Manage administration rights using admin views
Manage policy administrators using root roles
Manage actions and activities creators
Manage objects and views creators
Manage users and roles administrators of the policy
Manage root policy administrators
Manage administration permissions
Remove an admininistration permission
Add an admininistration permission
Manage administration roles
Add an administration role
Manage administration rights using admin roles
Back
Copyright © 2011 Entr'ouvert

Veridic

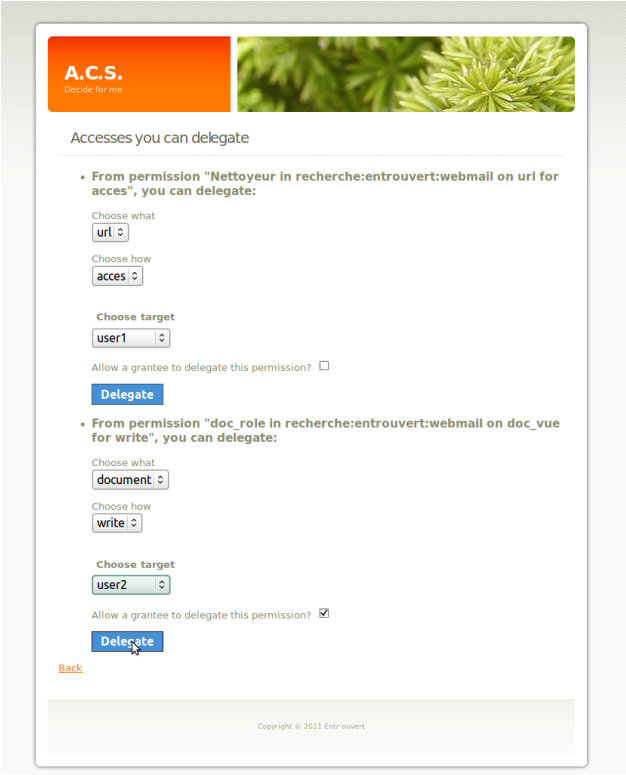


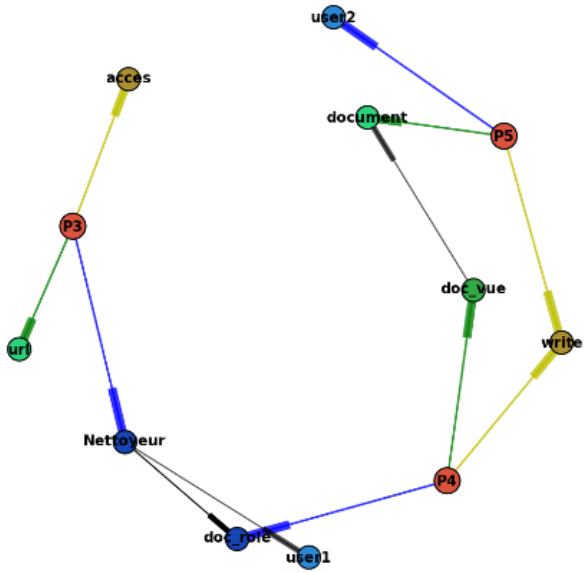


Veridic


A.C.S.
Decide for me
Access control policy
acces
Netrouteur
P3
P4
write
document
Back
Copyright © 2011 Entr'ouvert

Veridic





Veridic

Tous les objets de la politique sont des instances de modèle.

```
class Role(models.Model):
    '''Role'''
    name = models.CharField(max_length = 40)
    namespace = models.ForeignKey(Namespace, verbose_name = _('Namespace'),
        default = '1')

    users = models.ManyToManyField('UserAlias',
                verbose_name=_('users'), blank=True)
    roles = models.ManyToManyField('Role', symmetrical=False,
        verbose_name=_('roles'), blank=True)
```

- Role : isAuthorizedRBAC0(who, what, how)
- Role et Activité : isAuthorizedRBAC1(who, what, how)
- Role, Activité et Vue : isAuthorizedRBAC2(who, what, how)

Authentic2.1

Plateform de gestion d'identité centralisée à l'automne 2011

- Intégration de OAUTH, éventuellement UMA
- Intégration du module de contrôle d'accès
- Délivrance d'autorisations avec XACML/SAML
- Contrôle d'accès basé sur la consommation de certificats multiples
- Refonte du reverse proxy pour l'intégration du SSO et du contrôle d'accès à des applications existantes
- Autorité de certification X509 intégrée (alla certifi.ca)

Authentic2.1

Plateforme de gestion d'identité centralisée à l'automne 2011

1. Déploiement d'un Web SSO, d'un fournisseur d'identité, d'un fournisseur d'attributs, d'un hébergeur de services ou de données personnelles
2. Système de décision centralisé, pouvant notamment donner des accès sur certificats
3. Intégration d'un WebSSO ou d'un système de contrôle d'accès centralisé à des applications existantes avec un reverse proxy
4. Environnement utilisateur de gestion d'identité.

Conclusion

Qui pour héberger l'environnement utilisateur de gestion de son identité numérique et de ses données personnelles ?