IdP de Shibboleth vs CAS d'Apereo, « le meilleur des mondes possibles »

Ludovic Auxepaules
RENATER
23-25, rue Daviel
75013 Paris

Anass Chabli
RENATER
23-25, rue Daviel
75013 Paris

Résumé

Cette présentation a pour objectif de faire ressortir les possibilités offertes par les dernières versions des deux serveurs d'authentification les plus répandus dans la communauté Enseignement Recherche : l'IdP Shibboleth et le serveur CAS d'Apereo. Nous reviendrons tout d'abord sur les principes généraux de l'IdP3 et des serveurs CAS (5.x et 6.x). Nous aborderons ensuite leurs principales évolutions récentes. Enfin, nous identifierons leurs différences et leurs similitudes de fonctionnement : les deux peuvent gérer notamment les protocoles SAML et CAS, le recueil du consentement de l'utilisateur, l'authentification MFA, etc. De ces éléments se dégagera légitimement la question suivante : Puis-je mettre en œuvre une seule brique technique dans mon SI, c'est-à-dire soit un serveur CAS d'Apereo, soit un IdP Shibboleth qui fait tout ?

Nous apporterons des éléments de réponse à cette question à travers plusieurs cas d'utilisation et d'intégration de l'IdP3 Shibboleth et du serveur CAS, « seul et ensemble ».

Mots-clefs

Authentification, CAS, IdP, Shibboleth, SAML, OpenId-Connect, MFA, Intégration

1 Introduction

Depuis ces quatre dernières années, les systèmes d'authentification open-source, portés par le consortium Shibboleth pour l'IdP (*Identity Provider*) Shibboleth et la fondation Apereo[1] pour le serveur CAS (*Central Authentication Service*), ont beaucoup évolué. Le fournisseur d'identité IdP Shibboleth et le serveur d'authentification CAS sont très présents dans de nombreux systèmes d'information des membres et partenaires de la fédération Éducation-Recherche depuis plus d'une quinzaine d'années et répondent à deux besoins historiques. Les établissements utilisent le plus souvent l'IdP Shibboleth pour l'authentification à des ressources web externes en « point à point » ou ayant recours à des fédérations d'identité nationale (la Fédération Éducation-Recherche), internationale (l'inter-fédération EduGain), ou privées (dites fédérations locales). Le serveur CAS est quant à lui plus utilisé pour centraliser les authentifications sur des applications web internes à un établissement. CAS agit ainsi comme support à l'authentification unique et transparente ou SSO (*Single Sign On*) de ce même établissement.

Dans cet article, nous rappellerons tout d'abord des éléments de contexte relatifs aux protocoles historiquement supportés par l'IdP Shibboleth et le serveur CAS, la manière dont ces serveurs

1 La fondation Apereo est née en 2012 de la fusion des organisations Jasig (sponsorisant les projets CAS, uPortal…) et Sakai.

d'authentification ont été utilisés par les applications des établissements et dans le contexte spécifique des fédérations d'identité. Dans une seconde partie, nous décrirons les principales évolutions des dernières versions majeures de l'IdP Shibboleth et du serveur CAS. Nous illustrerons des pistes de réponse à la question *« Puis-je mettre en œuvre une seule brique technique dans mon SI, c'est-à-dire soit un serveur CAS d'Apereo, soit un IdP Shibboleth qui fait tout ? »* en nous focalisant sur différents cas d'utilisation et d'intégration de l'IdP3 Shibboleth et du serveur CAS, « seul et ensemble ». Nous compléterons ces réponses avec la présentation de nouvelles possibilités d'utilisation offertes par l'IdP Shibboleth et le serveur CAS vis-à-vis des nouveaux protocoles d'authentification et d'autorisation, de nouveaux usages en interrompant la phase d'authentification et de quelques fonctionnalités améliorant la sécurité lors de l'authentification.

2 Contexte historique et spécifique aux fédérations

2.1 SAML vs CAS

Les protocoles supportant le SSO dans l'IdP Shibboleth et le serveur CAS sont historiquement différents :

- Les normes **SAML (*Security Assertion Markup Language)* 1.0** (2002), **1.1** (2003) et **2.0** (2005) du consortium OASIS [1] sont supportés dans l'IdP Shibboleth et le SP (*Service Provider*[2]) Shibboleth. Seul SAML 1.1 était géré dans le serveur CAS sur les versions 3.x. ;
- Les protocoles **CAS 1.0** (2000) par l'Université de Yale, **2.0** (2005) et **3.0** (2014) de la fondation Apereo [2] sont supportés dans le serveur CAS.

À l'heure actuelle, seuls les protocoles SAML 2.0, CAS 2.0 et CAS 3.0 sont encore recommandés. La norme SAML est plus riche, plus complexe et plus « sécurisée » que le protocole CAS. Ils ont pour point commun d'être utilisables avec un mécanisme de redirections HTTP[3] entre le navigateur de l'utilisateur, l'application web cliente et le serveur d'authentification. Les réponses aux requêtes d'authentification sont exprimées en XML.

La délégation de l'authentification d'une application web cliente consiste en d'autres termes à :

- Faire confiance à la gestion de la vérification de l'identité de l'utilisateur par le serveur d'authentification ; accepter les affirmations dans les réponses du serveur : le fait que la connexion de l'utilisateur soit acceptée ou refusée et que les informations transmises sur la personne sont exactes ;
- Gérer techniquement les redirections HTTP et traiter les requêtes, les réponses, les assertions en XML afin que la cinématique d'authentification se poursuive et soit complète ;
- Mettre éventuellement en corrélation les informations retournées sur la personne par le serveur avec un profil géré localement dans le client.

Entre SAML et CAS, il existe deux différences notables, relatives à la sécurité et la confiance des clients avec les serveurs d'authentification et inversement :

- En SAML, un **enregistrement/échange préalable entre le SP et l'IdP est obligatoire**. En d'autres termes, afin qu'ils puissent communiquer, l'IdP doit connaître les métadonnées (l'identifiant de l'entité SAML : *entity-id*, les points de terminaison : *endpoints, ...*) décrivant le SP et le SP doit connaître les métadonnées de l'IdP. Les assertions des requêtes et réponses en XML sont le plus souvent **signées (et chiffrées) à l'aide de clés cryptographiques asymétriques**[4].

2 Dans la terminologie SAML, le client est nommé SP (*Service Provider)* et le serveur d'authentification est un IdP (*Identity Provider*). Le SP Shibboleth et l'IdP Shibboleth sont deux exemples de client et de serveur SAML.

3 Avec SAML, le navigateur peut jouer l'intermédiaire et il n'y aura pas alors de relation directe entre l'IdP et le SP. Ce n'est pas le cas de CAS où la validation du *Service Ticket* est réalisée entre le client et le serveur.

– Avec CAS, a**ucun enregistrement préalable des applications n'est nécessaire au sein du serveur d'authentification** : tout service web peut appeler les points d'accès de *login* et de *logout* du serveur CAS. La confiance est établie entre le client et le serveur CAS au niveau de la validation du certificat https du serveur CAS par le client (pas dans le contenu des données échangées). Cependant, il est recommandé de mettre en place un **filtrage des services[5] qui sont autorisés** à communiquer avec le serveur CAS.

2.2 « SAMLiser » / « Shibboliser » vs « cassifier »

Pour profiter du SSO et fonctionner avec CAS ou Shibboleth, une application web doit être le plus souvent adaptée afin d'être capable de déclencher la connexion (*login*), la déconnexion (*logout*), d'exploiter des attributs utilisateur (à minima un identifiant), etc. Le travail d'adaptation des applications à réaliser n'est pas exactement le même selon les cas : on parle de « cassification » (ou de « cassifier » une application) dans le contexte de la communication avec un serveur CAS, de « samlisation » pour la communication avec un IdP Shibboleth et plus particulièrement de « shibbolisation » (ou « shibboliser » une application) lorsqu'un SP Shibboleth est mis en place pour gérer l'authentification SAML à la place de l'application [3].

De par la simplicité du protocole CAS, la « cassification » est réalisée le plus souvent directement dans les applicatifs en ayant recours à des bibliothèques clientes officielles (phpCAS, java-cas-client, dotnet-cas-client, mod_auth_cas), à des frameworks supportant CAS (pac4j, spring security) ou à des intégrations directement dans l'application.

À l'image de la « cassification », une application peut être modifiée pour être rendue compatible avec SAML. Il existe des briques logicielles dans différents langages et frameworks (SimpleSAMLphp, spring-security…). Néanmoins, une des approches les plus connues consiste à « shibboliser » une application existante. Sur Linux, le SP Shibboleth est composé de deux parties : un module dans le serveur web apache (*mod_shib)* et un serveur autonome (*shibd*). L'application « shibbolisée » se charge de rediriger l'utilisateur vers le SP Shibboleth puis de récupérer son identifiant et ses attributs, soit à partir de variables d'environnement, soit à partir de champs d'entêtes HTTP. Un SP Shibboleth peut être mutualisé pour plusieurs applications. Il peut être configuré tel un reverse-proxy au travers duquel passeront toutes les requêtes.

2.3 Quelques spécificités liées aux fédérations d'identité

Dans un contexte où des applications sont amenées à déléguer leur authentification à de nombreux serveurs de tiers dédiés, il devient difficile de gérer toutes les relations de gré à gré. Dans le contexte international de l'enseignement supérieur et de la recherche, les fédérations d'identité facilitent la mise en place de ces relations notamment en utilisant la norme SAML 2.0. Elles évitent ainsi que chacun des participants n'ait à enregistrer et maintenir manuellement ces relations chacun de son côté : l'opérateur de fédération se charge d'agréger et de publier les métadonnées des différentes entités SAML (SP et IdP) publiquement (ou de manière privée pour les fédérations dites locales) afin que les SP et les IdP puissent établir une relation de confiance mutuelle et communiquer via des échanges sécurisés.

Dans le cas de RENATER, la publication est réalisée sous forme de **fichiers de métadonnées SAML** qui recensent toutes les informations techniques sur les fournisseurs d'identité (IdP) et les fournisseurs de service (SP). Ces informations sont renseignées par les responsables des IdP et des SP au travers du guichet de la fédération [4]. Pour chaque fédération, un premier fichier publie l'ensemble des SP et un second fichier publie l'ensemble des IdP. Il est nécessaire que ces fichiers soient téléchargés

4 Ces clés publiques asymétriques sont publiées dans les métadonnées respectives de l'IdP et du SP. Le SP utilise la/les clés publiques de l'IdP concernées pour signer et chiffrer les assertions et les données ; l'IdP vérifie la signature et déchiffre les données avec ses clés privées. L'IdP signe les assertions et chiffre les données avec les clés publiques du SP et le SP vérifie la signature et déchiffre les données avec ses clés privées.

5 On parle ici du *service registry* de CAS correspondant le plus souvent à un ou plusieurs fichiers XML ou Json. Ces fichiers décrivent les services (clients) plus ou moins finement et notamment les autorisations ou interdictions sous forme d'expressions régulières relatives aux URL ou IP source des services.

régulièrement par les IdP et les SP. Des fédérations étrangères (*Incommon* en premier) commencent à mettre en place en complément des services plus flexibles de distribution des métadonnées : ils sont basés sur le protocole MDQ (*Metadata Query Protocol*) [5] qui permet de télécharger dynamiquement les métadonnées d'une seule entité SAML au moment de l'authentification. Ce nouveau mode de diffusion des métadonnées évite notamment de télécharger l'intégralité des métadonnées en une fois à plusieurs moments de la journée.

Les fédérations d'identité ajoutent une complexité supplémentaire pour les SP fédérés au moment de l'authentification de l'utilisateur : en effet, lorsqu'il est sollicité par une application fédérée, le SP ne peut pas savoir directement avec quel fournisseur d'identité l'utilisateur va être amené à se connecter. SAML normalise ainsi le protocole DS (*Discovery Service)*[6] qui permet de placer un **service de découverte** des IdP au niveau d'un SP : un DS liste les différents IdPs autorisés à communiquer avec le SP. Le navigateur de l'utilisateur est aiguillé vers le SP et l'IDP par une suite de redirections HTTP. Par exemple, RENATER met à disposition un service de découverte pour chacune des trois fédérations publiques qu'il opère [6].

Les fédérations d'identité impliquent certaines recommandations sur les attributs de personnes fournis afin de normaliser les échanges entre les SP et les IdP concernant le nom, le format et le contenu des attributs demandés. Par exemple, pour ceux relatifs à l'enseignement supérieur et à la recherche en France, le référentiel international eduPerson [7] complété par supann [8] précise les différentes recommandations à suivre pour les annuaires des établissements. Les exemples ci-dessous présentent les recommandations de deux attributs communément utilisés dans le contexte des fédérations :

- ***eduPersonPrincipalName*** : identifiant globalement unique pour une personne, composé de deux parties séparées par un "@": celle de gauche identifie la personne au sein d'un établissement et celle de droite identifie cet établissement (le nom de domaine de l'établissement en général) ;
- ***eduPersonTargetedID*** : un identifiant de personne globalement unique, opaque, pérenne, non réattribuable qui a une valeur fournie différente pour chaque SP visité.

De plus, il est recommandé aux IdP de ne fournir que les informations nécessaires et suffisantes aux SP et de respecter la liste des attributs obligatoires pour chaque SP (diffusés sous forme de filtre des attributs ou directement exprimées dans les métadonnées du SP). Une automatisation de ce filtrage est souhaitable pour le bon fonctionnement des SP fédérés.

3 Les évolutions de l'IdP 3.x et de CAS 5.x et 6.x

La fin de vie des deux solutions historiques majoritairement utilisées dans l'enseignement supérieur et la recherche – à savoir la version 2.x. de l'IdP Shibboleth en 2016 et la version 3.x du serveur CAS en 2015 – pousse les établissements à mettre à jour leurs différents systèmes d'authentification vers des versions plus récentes et encore maintenues par la communauté. Ces mises à jour s'inscrivent quelquefois dans des projets plus larges de modernisation des systèmes d'information et dans le contexte général de la gestion des identités et des accès (IAM).

Les projets open-source Shibboleth et CAS bénéficient tous les deux de communautés d'utilisateurs et développeurs dynamiques qui répondent le plus souvent aux questions posées aussi bien par rapport à des nouveaux cas d'usage ou expérimentaux que par rapport à des cas d'usage plus éprouvés. Un travail important d'évolution a été mené dans les dernières versions de ces deux briques techniques.

6 Le DS est pratiquement équivalent fonctionnellement au WAYF (*Where Are You From)?* qui a été défini seulement par les développeurs de Shibboleth et est spécifique au SP Shibboleth.

3.1 CAS 5.x et 6.x

L'avènement des versions majeures successives du CAS d'Apereo (CAS 4.x, 5.x et 6.x) a complètement bouleversé la manière de configurer et d'intégrer un serveur CAS. Ce dernier est ainsi passé progressivement d'un serveur utilisé essentiellement comme *Single Sign On* web via le protocole d'authentification CAS (1.0, 2.0) à un système modulaire permettant d'intégrer de nombreuses fonctionnalités.

- Le **protocole d'authentification CAS 3.0** sorti en 2015 a officialisé la possibilité de transmettre lors de la phase d'authentification des attributs relatifs à la personne connectée (nom, prénom, mail…) en plus de l'identifiant ;
- Le serveur CAS peut s'adapter à de **nombreuses sources d'authentification** (LDAP, fichiers, JDBC, x509, Cassandra, Radius, Kerberos, JWT, REST...) ou approvisionner les attributs de personne, de **nombreux systèmes de stockage** (Memcached, Redis, Oracle, Hazelcast, CouchBase, MongoDB…) ;
- CAS est capable de jouer le rôle de **serveur d'authentification** (IdP SAML2, fournisseur oAuth, fournisseur OpenId-Connect, serveur CAS) mais il peut être tout aussi bien **un client délégant l'authentification à un autre serveur** : un IdP SAML, ADFS, Facebook, Google, Twitter, un autre serveur CAS, etc. ;
- Le serveur CAS propose des intégrations pré-configurées avec de nombreux services et applications en ligne (*Google Apps, Microsoft Office 365, Gitlab, Slack, Zoom, Zimbra, Jira…)* et également dans le contexte de l'**authentification renforcée ou MFA** (*Yubikey, Duo Securiy, Authy, Google Authenticator, Simple MFA…*).

L'ensemble des configurations, dont les **politiques de filtrage des attributs**, peuvent être définies de manière générale dans le fichier principal de configuration de CAS (*cas.properties*) et surchargées ou complétées directement pour un service ou un ensemble de services donnés à l'aide du *service registry*. L'implémentation historique sous forme de fichiers XML tend à disparaître pour laisser place à une description plus simple sous forme de fichiers JSON.

Un exemple de contenu d'un fichier JSON décrivant un « service CAS » est reporté ci-dessous. Les services autorisés à utiliser le serveur CAS correspondent à des URL conformes à l'expression régulière définie dans le champ *serviceId*. Ils participent au SSO (*ssoEnabled* à *true*), n'utilisent ni le module de recueil des consentements, ni le mode *Proxy* du protocole CAS. Le client CAS reçoit, à l'issue de l'authentification, tous les attributs de l'utilisateur produits par CAS à partir de toutes les sources de données, à l'exception du mot de passe et du ticket de proxy (*authorizedToReleaseCredentialPassword* et *authorizedToReleaseProxyGrantingTicket* à *false).* Enfin, après l'authentification principale avec un couple identifiant mot de passe, un second facteur d'authentification sera exigé systématiquement à l'utilisateur connecté pour accéder à l'application : ici *mfa-simple* est un mécanisme simple de MFA intégré directement au serveur CAS qui supporte l'envoi d'un jeton par mail ou sms (en fonction de la configuration définie globalement dans le fichier *cas.properties*).

```
{ "@class" : "org.apereo.cas.services.RegexRegisteredService",
"serviceId" : "^https://xxxx[.]univ[.]fr(/app)?/.*?",
"name" : "Accès sécurisé à xxx",
"id" : 1010,
"description" : "Interface de gestion sécurisée de l'application xxx",
"proxyPolicy" : {
  "@class" : "org.apereo.cas.services.RefuseRegisteredServiceProxyPolicy"
},
"evaluationOrder" : 1010,
"attributeReleasePolicy" : {
  "@class" : "org.apereo.cas.services.ReturnAllAttributeReleasePolicy",
```

```
  "authorizedToReleaseCredentialPassword" : false,
  "authorizedToReleaseProxyGrantingTicket" : false },
"consentPolicy": {
  "@class":
  "org.apereo.cas.services.consent.DefaultRegisteredServiceConsentPolicy",
  "enabled": false },
"multifactorPolicy" : {
"@class" :
  "org.apereo.cas.services.DefaultRegisteredServiceMultifactorPolicy",
  "multifactorAuthenticationProviders" :
    [ "java.util.LinkedHashSet", [ "mfa-simple" ] ]  },
"accessStrategy" : {
  "@class" :
  "org.apereo.cas.services.DefaultRegisteredServiceAccessStrategy",
  "enabled" : true,
  "ssoEnabled" : true
} }
```

L'architecture générale du serveur CAS et la communication avec ses clients est illustrée ci-dessous :

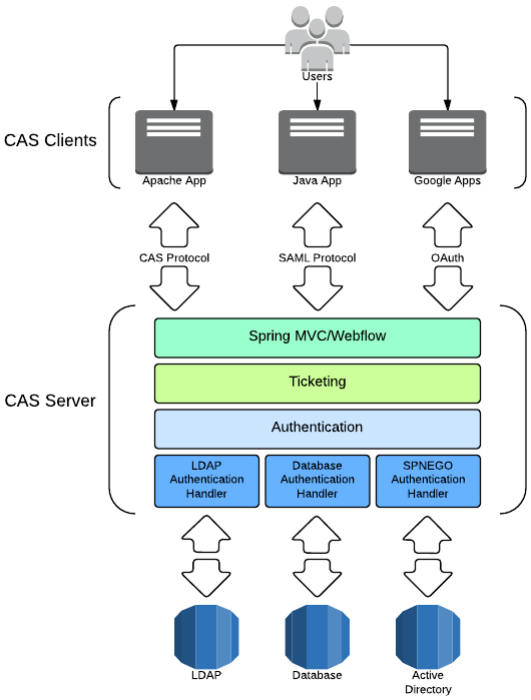

Figure 1 - Architecture générale du serveur CAS 6.x [9]

Le projet international CAS d'Apereo est passé progressivement d'un écosystème avec un projet principal gérant quelques fonctionnalités et étendu par des contributions et des extensions suivies de manière hétérogène par la communauté, à un projet totalement réécrit, très modulaire. Il a ainsi été enrichi par de nombreuses fonctionnalités activables par une configuration de propriétés de type « clé-valeurs » (l'intégration d'une dépendance et configuration d'une ou plusieurs lignes dans un fichier de propriétés). Plusieurs centaines de fonctionnalités et modules sont déjà accessibles dans le projet principal de CAS. Néanmoins, le projet CAS évoluant très/trop vite, il peut subsister des bugs, des fonctionnalités implémentées partiellement ou assez peu documentées… Il est ainsi souvent nécessaire qu'un ou

plusieurs « membres pionniers » de la communauté s'intéressent à la mise en place d'un module pour que ce dernier devienne complètement fonctionnel pour un usage en production.

Un exploitant peut assez facilement installer un serveur CAS s'il ne sort pas des usages courants et bien documentés du projet et s'il prend en compte les retours d'expériences de la communauté. Cependant la mise en œuvre d'un serveur CAS intégrant de nombreuses fonctionnalités reste toujours une activité assez orientée vers un public de « développeurs » : ces derniers auront la possibilité de « chercher dans le code source », interpréter les exceptions en Java et corriger certains problèmes, puis de partager leurs développements avec la communauté internationale.

3.2 IdP Shibboleth 3.x

Le projet IdP de Shibboleth a évolué lui aussi, notamment avec la sortie des versions 3.x dès la fin 2015. Les principaux objectifs de cette version sont de **faciliter le développement d'extensions** en les rendant moins complexes et plus standards. Cette nouvelle organisation du projet Shibboleth vise également à découpler l'IdP du protocole SAML afin de faciliter son **adaptation à d'autres protocoles d'authentification et d'autorisation** (tel qu'OpenId-Connect).

Les différents fichiers de configuration ont été ainsi réorganisés pour mieux cibler les fonctionnalités, les rendre plus lisibles. Ils peuvent dorénavant être modifiés « à chaud » sans avoir à relancer l'IdP. Un fichier de configuration principal de l'IdP a été introduit (*idp.properties*) ainsi que plusieurs fichiers pour gérer directement certaines fonctionnalités :

- *ldap.properties* pour gérer la configuration des sources de données liés au LDAP lors de l'authentification ;
- *metadata-providers.xml* pour gérer la configuration des sources de métadonnées SAML.

Les fichiers définissant l'**approvisionnement des attributs utilisateurs** à partir d'une ou plusieurs sources de données ainsi que le fichier de **configuration des politiques de diffusion des attributs** aux SP sont toujours présents (respectivement *attribute-resolver.xml* et *attribute-filter.xml*). Cette manière de gérer les attributs et les SP est performante dans le contexte des fédérations d'identité.

L'exemple suivant illustre deux politiques de filtrage : la première consiste à fournir à tous les SP présents dans la Fédération Éducation-Recherche (*groupID*), une liste d'attributs demandés comme étant obligatoires (*onlyIfRequired="true"*) par le SP. La seconde règle consiste à fournir certains attributs prédéfinis pour les SP appartenant à la catégorie *Research and Scholarship* (R&S)[7] [10].

```
<AttributeFilterPolicyGroup id="FilterPolicy"
    xmlns="urn:mace:shibboleth:2.0:afp"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="urn:mace:shibboleth:2.0:afp
    http://shibboleth.net/schema/idp/shibboleth-afp.xsd">

<AttributeFilterPolicy id="releaseToRenaterProdSPs">
  <PolicyRequirementRule xsi:type="InEntityGroup"
      groupID="https://federation.renater.fr/" />
  <AttributeRule attributeID="eduPersonTargetedID">
    <PermitValueRule xsi:type="AttributeInMetadata" onlyIfRequired="true"/>
  </AttributeRule>
  <AttributeRule attributeID="mail">
      <PermitValueRule xsi:type="AttributeInMetadata" onlyIfRequired="true"/>
  </AttributeRule>
  <AttributeRule attributeID="eduPersonPrincipalName">
      <PermitValueRule xsi:type="AttributeInMetadata" onlyIfRequired="true"/>
  </AttributeRule>
```

7 La liste des attributs recommandés pour R&S est *eduPersonPrincipalName, eduPersonTargetedID, email, displayName, givenName, surname, eduPersonScopedAffiliation.*

```
  …
</AttributeFilterPolicy>

<AttributeFilterPolicy id="releaseResearchAndScholarship">
  <PolicyRequirementRule xsi:type="EntityAttributeExactMatch"
      attributeName="http://macedir.org/entity-category"
      attributeValue="http://refeds.org/category/research-and-scholarship"/>
  <AttributeRule attributeID="eduPersonPrincipalName">
    <PermitValueRule xsi:type="ANY"/>
  </AttributeRule>
  <AttributeRule attributeID="eduPersonTargetedID">
    <PermitValueRule xsi:type="ANY"/>
  </AttributeRule>
  <AttributeRule attributeID="mail">
    <PermitValueRule xsi:type="ANY"/>
  </AttributeRule>
  …
</AttributeFilterPolicy>
</AttributeFilterPolicyGroup>
```

De nouvelles fonctionnalités ont également été ajoutées directement dans le projet principal de l'IdP :

- un **module de recueil de consentement des utilisateurs ;**
- une **implémentation du protocole CAS ;**
- un meilleur découpage des pages pour faciliter leur personnalisation ;
- le **support de l'authentification renforcée (MFA) ;**
- l'introduction de nouveaux profils SAML.

Le SLO (*Single Logout*) est mieux géré dans les IdP 3.x : si les sessions sont gérées sur le serveur, l'IdP peut ainsi propager des requêtes de déconnexion au moins à l'ensemble des SP qui supportent la déconnexion par réception d'une requête HTTP sur un point de terminaison décrit dans leurs métadonnées.

L'architecture suivante fait ressortir la manière dont l'IdP Shibboleth traite les requêtes, contrôle les profils SAML utilisés par le SP et l'enchaînement des appels des composants et des systèmes externes :

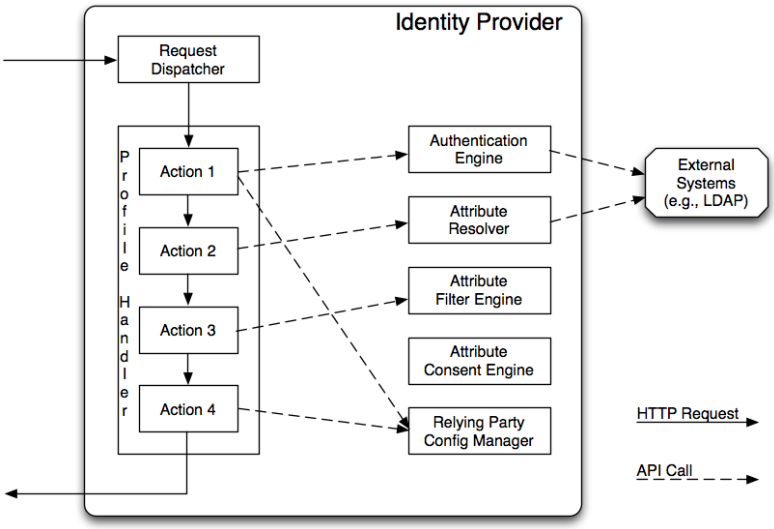

Figure 2 - Architecture de l'IdP Shibboleth 3.x [11]

Le projet de l'IdP Shibboleth a l'avantage d'être bien documenté, de décrire une liste exhaustive des modifications, des fonctionnalités dépréciées à chaque changement de version (ce qui facilite les montées de versions mineures). Il présente assez peu de bugs, mais l'ajout de fonctionnalités avancées reste encore le plus souvent sous forme d'extensions et de contributions de la communauté [12] dissociées du projet principal de l'IdP.

Les extensions et contributions ont un cycle de vie différent de celui du projet de l'IdP Shibboleth. Elles nécessitent le plus souvent d'ajouter, de modifier ou de surcharger des fichiers ou répertoires. Par exemple, l'université de Chicago a expérimenté l'ajout du support du protocole *OpenID-Connect* dans un IdP 3. Elle a mis à disposition l'ensemble des modifications dans un projet nommé *shibboleth-oidc* compatible avec un IdP 3.2. Il n'est malheureusement ni garanti que cette intégration soit complètement opérationnelle, ni qu'elle restera fonctionnelle.

Si la configuration des sources de données dans l'IdP est insuffisante ou inadaptée à certains cas d'utilisation spécifiques (par exemple pour gérer des mots de passe stockés en base de données avec les algorithmes BCRYPT, Argon2…), il est toujours possible de développer et d'intégrer une nouvelle extension basée sur JAAS.

Même si l'IdP 3 Shibboleth reste toujours assez complexe à configurer[8] pour un novice, notamment lors des montées de versions majeures (passage d'un IdP 2 à un IdP 3 avec les nombreux changements de syntaxe, disparition/remplacement de certaines fonctionnalités…), il peut être mise en place par un exploitant n'ayant pas nécessairement de compétences en développement si les besoins sont « classiques » (utilisation d'un annuaire LDAP comme source de données et configuration décrite par les différents opérateurs de fédération).

Dans des cas plus complexes, il est recommandé de posséder de bonnes connaissances de la norme SAML 2.0 et du fonctionnement interne de l'IdP Shibboleth afin d'approfondir sa configuration en surchargeant certains éléments, « scriptant » certaines parties, voire en développant des extensions ou des composants manquants (et dans l'idéal les partager à la communauté afin qu'ils puissent être réutilisés ou même inspirer les développeurs de l'IdP Shibboleth).

4 Différents cas d'utilisation de l'IdP Shibboleth et de CAS

À partir des différents éléments exposés précédemment se dégage légitimement la question suivante :

> *Puis-je mettre en œuvre une seule brique technique dans mon SI, c'est-à-dire soit un serveur CAS d'Apereo, soit un IdP Shibboleth qui fait tout ?*

Dans cette partie, nous apportons des éléments de réponse à cette question à travers plusieurs cas d'utilisation et d'intégration de l'IdP3 Shibboleth et du serveur CAS 5.x ou 6.x, « seul et ensemble » :

- Utilisation d'une seule brique technique pour « tout faire » :
  - IdP Shibboleth utilisé en tant que serveur CAS ;
  - Serveur CAS utilisé en tant qu'IdP SAML.
- Utilisation des deux briques ensemble qui consiste à déléguer l'authentification de l'un à l'autre :
  - « Cassifier » l'IdP Shibboleth (Shibboleth en client CAS) ;
  - « Shibboliser » CAS (CAS en mode SP SAML).

8 La configuration de nombreux fichiers XML pour mettre en place les différentes fonctionnalités de l'IdP. De plus, il est assez souvent difficile de comprendre certaines exceptions en Java en cas d'erreur de configuration et du fait de la complexité de certains profils SAML.

4.1 Utilisation d'une seule brique technique pour « tout faire »

Nous avons constaté précédemment que l'IdP Shibboleth et le serveur CAS d'Apereo sont capables de gérer les deux protocoles SAML et CAS. Dans cette partie, nous ne revenons pas sur les utilisations historiques et bien connues de l'IdP Shibboleth en tant que fournisseur d'identité SAML et du serveur CAS avec le protocole CAS. Nous discutons des cas d'utilisation consistant à mettre en place une des briques pour remplacer l'autre. Nous faisons notamment ressortir certains avantages, différences de configuration et limitations.

4.1.1 IdP Shibboleth utilisé en tant que serveur CAS

L'IdP 3.x Shibboleth est capable de gérer le protocole CAS 2.0 et les fonctionnalités avancées de *proxy tickets, gateway, renew* et la transmission d'attributs avec le protocole SAML 1.1. Les points d'accès */login*, */serviceValidate*, */proxyValidate*, */samlValidate* sont activables dans l'IdP Shibboleth afin de gérer le protocole CAS [13]. Il n'y a pas de gestion de point d'accès spécifique */logout* car Shibboleth ne gère pas la déconnexion de la même manière que CAS ; cette limitation impose donc d'utiliser le mécanisme de SLO de Shibboleth. Les sessions et les tickets relatifs au protocole CAS peuvent être gérés soit en mémoire, soit avec *Memcached* ou encore dans base de données. Les services autorisés à utiliser le protocole CAS peut être déclarés dans le fichier *cas-protocol.xml*, dans un format proche de celui du *service registry* historique de CAS (avec des expressions régulières sur les URL des services). L'exemple suivant illustre la déclaration des services autorisés à utiliser le protocole CAS dans l'IdP Shibboleth. La priorité de traitement des services est réalisée par ordre de définition dans la liste.

```
<bean id="cas.serviceRegistry"
    class="net.shibboleth.idp.cas.service.PatternServiceRegistry">
  <property name="definitions">
   <list>
    <bean class="net.shibboleth.idp.cas.service.ServiceDefinition"
      c:regex="https:\/\/xxxx\.univ\.fr(:\d+)?\/.* ?"
      p:group="xxx-services"
      p:authorizedToProxy="false" />
   </list>
  </property>
</bean>
```

L'IdP Shibboleth utilisé avec le protocole CAS est ainsi compatible directement avec de nombreux clients CAS. Cependant, la dernière version du protocole CAS (i.e. 3.0) n'est pas implémentée : une version modifiée du protocole CAS 2.0 permet de récupérer en plus de l'identifiant une liste d'attributs de l'utilisateur connecté (ce mode de fonctionnement était communément utilisé par exemple par les applications reposant sur le client phpCAS). Il faut donc rester conscient que tous les cas de figure offerts par le protocole CAS 3.0 peuvent ne pas être gérés et des dysfonctionnements avec certains clients récents sont possibles.

En faisant un parallèle avec le serveur CAS, l'IdP 3.x est quasi-équivalent fonctionnellement à un serveur CAS 3.x intégrant la gestion des protocoles SAML 1.1 et CAS 2.0. La déclaration des services y étant assez similaire, assez peu d'efforts sont nécessaires pour passer de l'un à l'autre.

Au fur et à mesure des nouvelles versions mineures de l'IdP, des bugs ou limitations sont corrigés et de nouvelles fonctionnalités sont intégrées. En 2018, l'IdP 3.4 apporte une nouvelle manière de décrire les services déclarés pour CAS dans l'IdP Shibboleth. Cette nouvelle notation est basée sur des métadonnées SAML. Différents éléments de configuration peuvent être décrits dans les métadonnées d'un service CAS : une ou plusieurs adresses d'accès au service CAS, les adresses de retours et les certificats si le mode Proxy est utilisé, la participation ou non au *Single Logout.* L'utilisation de la notation SAML est

intéressante, car elle permet de faire un pont entre le monde SAML et CAS en décrivant de manière plus standardisée les services cassifiés. Ceci pourrait permettre de les partager tout comme les métadonnées SAML des IdP et des SP.

4.1.2 Serveur CAS utilisé en tant qu'IdP SAML

Un travail important a été réalisé au niveau du serveur CAS 4.x, 5.x et 6.x afin qu'il puisse gérer l'authentification SAML 2.0 en tant qu'IdP [14]. L'intérêt principal réside dans la facilité de configuration de toute la partie SAML2 pour activer l'IdP du serveur CAS. En effet, il est nécessaire seulement d'intégrer le module *cas-server-support-saml-idp* et ensuite de configurer quelques lignes dans le fichier *cas.properties* (au moins l'*entity-id*, le *scope* et l'emplacement des métadonnées de l'IdP) :

```
cas.authn.samlIdp.entityId=https://casserver.univ.fr/cas/idp
cas.authn.samlIdp.scope=@univ.fr
cas.authn.samlIdp.metadata.location=file:/etc/cas/config/saml
```

À partir de là, les métadonnées SAML de l'IdP vont être générées dynamiquement si elles n'existent pas au premier appel du point d'accès */idp/metadata*. La liaison entre l'IdP de CAS peut être réalisée très rapidement en « point à point » avec différents SP (i.e. en dehors d'une fédération d'identité) : pour cela, il est nécessaire de renseigner les métadonnées de l'IdP dans les SP concernées. Du côté de l'IdP de CAS, les métadonnées de chaque SP sont déclarables sous forme d'un nouveau fichier JSON du *service registry* ; le *service-id* du nouveau service autorisé à utiliser CAS est l'*entity-id* du SP.

L'exemple suivant illustre la déclaration d'un SP autorisé à utiliser l'IdP de CAS dont les métadonnées SAML et la clé de signature ont été installées localement. Les attributs retournés par l'IdP sont décrits explicitement dans le service ainsi que les *friendly-name* associés. Il est précisé également que les assertions sont signées et les données chiffrées.

```
{
 "@class" : "org.apereo.cas.support.saml.services.SamlRegisteredService",
 "serviceId" : sp-entityid",
 "name" : "Test SP",
 "id" : 10004, "description" : "Test SP en SAML2 (mode IdP de CAS)",
 "metadataLocation" : "/etc/cas/saml/sps/sp-metadata.xml",
 "metadataSignatureLocation" : "/etc/cas/saml/sps/sp-public-key.crt",
 "signAssertions" : true,
 "signResponses" : true,
 "encryptAssertions" : true,
 "signingCredentialType" : "X509",
 "attributeReleasePolicy" : {
   "@class" : "org.apereo.cas.services.ReturnAllowedAttributeReleasePolicy",
   "allowedAttributes" : [ "java.util.ArrayList",
     [ "givenName", "mail", "sn", "eduPersonPrincipalName" ]
   ] },
 "attributeFriendlyNames": { "@class": "java.util.HashMap",
   "urn:oid:1.3.6.1.4.1.5923.1.1.1.6": "eduPersonPrincipalName",
   "urn:oid:2.5.4.42": "givenName",
   "urn:oid:2.5.4.4": "sn",
   "urn:oid:0.9.2342.19200300.100.1.3": "mail" },
 "evaluationOrder" : 10004
}
```

De plus, il est possible de lier plus simplement l'IdP de CAS à différents SP et d'applications dont la configuration a déjà été automatisée dans CAS. Dans ce cas de figure, il est nécessaire d'activer le module *cas-server-support-saml-sp-integrations* et de renseigner ensuite la configuration du SP sous forme de « clés-valeurs » dans le fichier *cas.properties*.

Autant il est plutôt simple d'intégrer les métadonnées ou les politiques de filtrage des attributs pour chaque SP une à une, autant le passage à l'échelle des fédérations d'identité peut se révéler un peu plus complexe avec le mécanisme de gestion individuelle des services dans le *service registry* du serveur CAS. En effet, la multiplication des fichiers de configuration, leur ordre d'évaluation et la redéfinition des politiques et des éléments de configuration par service ne facilitent pas la lisibilité globale. Des *entity-id* multiples, sous forme d'expressions régulières, sont exprimables dans le champ *serviceId* d'un service (tout comme les services cassifiés). Une clé publique de signature doit être définie afin de valider la signature des métadonnées.

La résolution des attributs réalisée dans l'*attribute repository* du serveur CAS (essentiellement par *mapping*) doit être cohérente et complète vis-à-vis des politiques de filtrage des attributs pour les différents SP (i.e. les attributs doivent au moins porter le nom et avoir le contenu attendu). Il est possible de générer, « scripter » certains attributs pour certains services, générer les identifiants de type *name-id* volatiles ou opaques tels que l'*eduPersonTargetId* ou le *transient-id*.

L'expression des politiques de filtrage des attributs est riche dans CAS. Ces politiques peuvent être chaînées entre elles pour permettre de traiter plusieurs règles (avec la classe *ChainingAttributeReleasePolicy*). L'ordre d'invocation des règles est le même que celui de définition dans le service lui-même. Des travaux ont été également amorcés pour automatiser les configurations spécifiques aux fédérations et à l'alimentation dynamique des métadonnées SAML avec MDQ [5]. Le service suivant permet par exemple de renvoyer les attributs recommandés pour les SP inscrits dans les fédérations américaines *Incommon* ou dans la catégorie *Research and Scholarship* (déjà présentée dans la partie 3.2 pour l'IdP Shibboleth) :

```
{
 "@class": "org.apereo.cas.support.saml.services.SamlRegisteredService",
 "serviceId": ".+",
 "name": "SPs de la fédération Incommon",
 "id": 3,
 "evaluationOrder": 10000,
 "metadataLocation": "https://url-metadata/metadata-aggregate-file.xml",
 "attributeReleasePolicy": {
   "@class": "org.apereo.cas.services.ChainingAttributeReleasePolicy",
   "policies": [
     "java.util.ArrayList", [
      { "@class":
        "org.apereo.cas.support.saml.services.InCommonRSAttributeReleasePolicy" },
      { "@class":
         "org.apereo.cas.support.saml.services.RefedsRSAttributeReleasePolicy" }
     ]
   ]}
}
```

Les développeurs du projet CAS améliorent l'intégration et la configuration de l'authentification SAML à chaque mise à jour et publient également différents exemples de configuration pour les versions CAS 6. La version 6.1 sortira en octobre 2019 et apportera de nombreuses améliorations dont la **gestion dynamique des métadonnées SAML2 avec le protocole MDQ**, **une gestion du chiffrement et de la signature des assertions SAML par service**, des correctifs au niveau de la gestion des *friendly-name* des attributs, des identifiants, l'ajout de points d'accès pour gérer les caches des métadonnées SAML dans CAS…

L'IdP du serveur CAS devient, au fur et à mesure des mises à jour, une alternative viable à l'IdP Shibboleth même dans un contexte de fédération. Sa configuration est cependant différente de l'IdP Shibboleth car très ancrée dans la logique et le fonctionnement du serveur CAS (l'absence de configuration des fichiers bien connus dans le monde Shibboleth : *attribute-resolver.xml* et *attribute-filter.xml*).

4.2 Utilisation des deux briques ensemble : délégation de l'authentification de l'une à l'autre

Cette partie expose les cas d'usage où l'IdP Shibboleth et le serveur CAS sont utilisés conjointement.

4.2.1 « Cassifier l'IdP Shibboleth » (Shibboleth en client CAS)

C'est l'intégration la plus connue et utilisée entre un IdP Shibboleth et un serveur CAS. Elle permet de déléguer dans l'IdP Shibboleth l'authentification à un serveur CAS externe (en général qui joue déjà le rôle de SSO pour les services internes d'un établissement). Dans ce contexte l'IdP Shibboleth joue le rôle de client CAS. Il récupère à l'issue de la phase d'authentification au moins l'identifiant de l'utilisateur connecté. À partir de cet identifiant, l'IdP peut rechercher dans une autre source de données (annuaire LDAP, base de données relationnelle) les attributs correspondant à la personne connectée et les communiquer au SP.

Une première implémentation de ce scénario consiste à mettre en place un client CAS sur un serveur web en amont de l'IdP qui se charge de renseigner la variable d'environnement REMOTE_USER à l'issue de la phase d'authentification. Cette variable est consommée par l'IdP Shibboleth en configurant le fichier spécifique *conf/authn/remoteuser-authn-config.xml*. En faisant ce choix, le couplage est faible et l'IdP Shibboleth ne connaît pas la nature du mode d'authentification en amont (les possibilités sont limitées).

Un scénario offrant meilleure intégration est possible en installant et configurant l'extension ***Shib-CAS*** d'Unicon dans l'IdP Shibboleth (*shib-cas-authn2* pour l'IdP 2 et *shib-cas-authn3* pour l'IdP 3). Le schéma suivant illustre le fonctionnement de l'IdP Shibboleth et du serveur CAS dans ce cas d'utilisation.

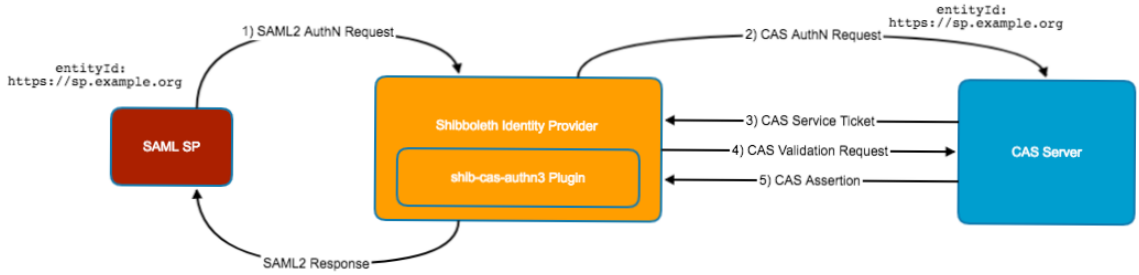

Figure 3 - Shib-CAS-authn3 : fonctionnement général lors d'une requête d'authentification [16]

La dernière version du module, **Shib-CAS-authn3** [17] apporte des nouveautés et renforce le couplage entre l'IdP Shibboleth et le serveur CAS. Le client CAS de l'IdP peut notamment utiliser le protocole CAS 3.0 qui permet de récupérer des attributs supplémentaires en plus de l'identifiant lors de la phase d'authentification.

De plus, tout comme en « mode IdP », CAS est capable également ici d'afficher sur sa page de connexion des informations sur le SP auquel l'usager souhaite se connecter. Ces informations sont extraites des métadonnées SAML *de la catégorie MDUI* [18]. Pour cela, l'IdP externe fournit l'*entity-id* du SP à CAS en paramètre de la requête. Le serveur CAS recherche ensuite dans les métadonnées des SP connus le nom, la description et l'image du service correspondant (à partir des métadonnées d'une fédération ou enregistrées localement en « point en point »). Les métadonnées sont récupérées à l'aide de l'activation du module *cas-server-support-saml-mdui* dont un exemple est fourni ci-dessous pour les fédérations Éducation-Recherche et de test :

```
## Displaying SAML MDUI with external IdP3
# cas.samlMetadataUi.requireValidMetadata=true
cas.samlMetadataUi.resources=https://metadata.federation.renater.fr/test/preview/
preview-sps-renater-test-metadata.xml::file:/etc/cas/saml/sps-metadata/renater-
metadata-signing-cert-2016.pem
# cas.samlMetadataUi.maxValidity=0
# cas.samlMetadataUi.requireSignedRoot=false
# cas.samlMetadataUi.parameter=entityId

## External Shibboleth IdP3 Integration
cas.authn.shibIdp.serverUrl=https://idpserver.fr/idp
```

Le module *Shib-CAS-authn3* est également compatible avec le nouveau profil *REFEDS MFA* [19] de l'IdP3 Shibboleth et permet de propager une demande d'authentification renforcée jusqu'au serveur CAS. L'avantage est de pouvoir potentiellement profiter des nombreux fournisseurs d'authentification MFA utilisables par configuration dans le serveur CAS. Les possibilités offertes par l'IdP Shibboleth et CAS au niveau du MFA sont décrites en détails dans un article que nous présentons aux JRES 2019 [20].

4.2.2 « Shibboliser CAS » (CAS en mode SP SAML)

Ce cas d'utilisation, un peu moins connu, est le cas opposé du précédent. Il permet de coupler un serveur CAS à un SP SAML afin de déléguer l'authentification à un ou plusieurs IdP externes. L'idée est ici de rendre possible rapidement une authentification via le protocole SAML 2.0 sur des applications qui sont déjà « cassifiées » et ne supportent pas ce protocole nativement, sans avoir ni à les modifier, ni à mettre en place un SP ou un *reverse-proxy SAML* en amont de chacune d'entre elles. Les modifications sont donc réalisées une seule fois au sein du serveur CAS qui intègre ou communique directement avec un SP.

Le projet open-source **CASShib,** créé par l'Université de Californie à Merced [21] et diffusé sous la licence Apache 2.0, a facilité la mise en place de ce cas d'usage sur des serveurs CAS 3.x[9] . Pour cela, CASShib surcharge la configuration d'un serveur CAS 3.x et active notamment le module *Trusted Authentication* de CAS pour qu'il lise l'identifiant de l'utilisateur connecté à partir des entêtes HTTP fournis par le SP Shibboleth. À partir de là, CAS initie une session pour l'utilisateur connecté, ainsi qu'un TGT (*Ticket Grant Ticket*) et un ST (*Service Ticket*) correspondant au service demandé. Enfin, CAS autorise l'accès à la ressource en fournissant le ticket ST généré. L'application cassifiée demande la validation du ticket ST auprès de CAS et redirige l'utilisateur vers la ressource demandée. Le traitement d'une requête d'authentification dans ce cas de figure est illustrée ci-dessous :

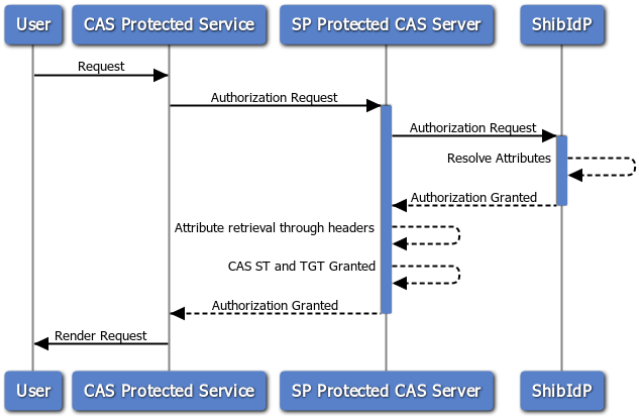

Figure 4 - Shibboliser CAS – traitement d'une requête d'authentification [23]

9 Un retour d'expérience a été présenté aux JRES 2015 dans le contexte du projet Toutatice [22].

CASShib n'est plus utilisable dans les versions modernes de CAS, mais il est toujours possible de configurer le module *Trusted Authentication* de CAS [24] de façon à ce que CAS puisse lire l'identifiant de l'utilisateur connecté depuis les entêtes HTTP fournies par le SP ou bien en l'extrayant des paramètres de la réponse relayée par le point d'accès */login* de CAS. Ce cas est décrit plus précisément sur le wiki de Shibboleth [23].

Une possibilité, plus moderne et plus simple à configurer, a été introduite dans CAS 5.x avec la rationalisation de toutes les fonctionnalités de délégation de l'authentification (et le recours au module de sécurité Pac4J [25]). CAS peut jouer directement le rôle de SP par configuration et être utilisable avec n'importe quel fournisseur d'identité SAML2 ; la mise en place d'une relation bilatérale (« point à point »)[10] est facilitée car les métadonnées du SP sont générées automatiquement en appelant la première fois le point d'accès */sp/metadata.* Ces métadonnées peuvent ensuite être enregistrées au niveau de l'IdP externe. Dans ce contexte, un bouton de connexion est ajouté à la page de login de CAS pour chaque IdP à qui il délègue l'authentification. Il n'y a pas de gestion d'un service de découverte (WAYF/DS) dans le serveur CAS en mode SP et il sera donc nécessaire de modifier les interfaces et la configuration du SP et la logique dans Pac4J si ce besoin est nécessaire (notamment dans un contexte de fédération).

5 Faire encore plus avec l'IdP Shibboleth et le serveur CAS

Même si l'IdP Shibboleth 3 supporte le protocole CAS et que le serveur CAS d'Apereo 5.x et 6.x apporte un support natif du protocole SAML 2.0, d'autres cas d'utilisation et possibilités peuvent orienter le choix d'utiliser l'un ou/et l'autre des deux serveurs d'authentification[11]. Dans cette partie, nous introduisons quelques-unes des possibilités offertes par l'IdP Shibboleth et le serveur CAS.

5.1 Gestion de nouveaux protocoles et délégation de l'authentification

Un des premiers enjeux des systèmes d'authentification actuels est d'être capable de gérer les nouveaux protocoles d'authentification et d'autorisation (tels que *OpenID-Connect*, oAuth2, *JSON Web Tokens*…) qui deviennent les « standards » de l'internet.

À l'aide des extensions et des contributions de la communauté [11] au projet Shibboleth, les authentifications via OpenId-Connect (par l'université de Chicago et Unicon) et les JWT (par l'université catholique de Louvain) peuvent être testées dans l'IdP Shibboleth. Néanmoins, il n'est pas prévu à l'heure actuelle qu'ils soient inclus au projet principal de l'IdP. L'intégration et les différents tests sont donc à réaliser soi-même pour pouvoir supporter ces protocoles et ensuite les adapter éventuellement à ses propres besoins. Cependant des travaux ont été amorcés par la communauté internationale afin de pouvoir proposer à l'avenir des **fédérations d'identité basées sur OpenId-Connect** [28] en complément de celles actuellement basées sur SAML 2.0.

Le projet CAS ayant choisi d'intégrer l'ensemble des fonctionnalités à son projet principal, il est plus simple de configurer et de tester des nouveaux protocoles : OpenId, OpenId-Connect, oAuth2, JWT… et de mener les adaptations à ses besoins. Le module Pac4j [25], intégré dans le cœur de CAS, lui offre également la capacité de se comporter comme **un « proxy/client universel ».** Il devient ainsi possible d'une part de s'authentifier à des systèmes d'authentification externes (IdP SAML2, fournisseurs d'Identité compatibles oAuth2 ou OpenId-Connect, ADFS) et d'autre part de communiquer avec les clients dans d'autres protocoles (CAS, SAML2, OpenId-Connect, JWT...). Cette fonctionnalité ouvre de grandes possibilités d'intégration à moindre coût en se servant de CAS comme intermédiaire avec des systèmes d'authentification externes au système d'information (par exemple France-Connect comme cela a été exposé dans un retour d'expérience en 2017 lors des Esup-Days n°23 [29]).

10 Un exemple plus complet de configuration est présenté sur le blog d'Apereo [24].

11 Ou tout autre solution alternative supportant les protocoles SAML et CAS tel que LemonLDAP::NG [27] pour n'en citer qu'une.

5.2 Interruption de l'authentification et apport de nouveaux usages

Mettre en pause ou interrompre la phase d'authentification avant l'accès à une application peut être intéressant dans le but de **présenter des informations à l'utilisateur** (sur l'état de son compte, sur l'état d'un service…) ou de réaliser une action avant de pouvoir poursuivre l'authentification (l'inviter à accepter une charte d'utilisation, changer son mot de passe expiré, renseigner une information manquante sur une plateforme externe). Dans leurs dernières versions, l'IdP Shibboleth et le serveur CAS supportent cette possibilité et peuvent ainsi être à même de gérer de nouveaux usages et fonctionnalités qui jusque-là étaient à la charge d'autres briques du SI. Trois cas principaux sont documentés aussi bien par CAS que Shibboleth :

- Le **recueil des consentements des utilisateurs** au niveau des attributs transmis à l'application ;
- La **gestion des mots de passe arrivant à expiration** ;
- L'**acceptation d'une charte d'utilisation ou de conditions générales d'utilisation**.

L'usage le plus éprouvé et le mieux documenté est le recueil des consentements de l'utilisateur (qui devient nécessaire avec le RGPD) avant de transmettre les données le concernant à l'application ayant initiée l'authentification. L'utilisateur peut visualiser la liste des attributs utilisateur qui seront transmis et leur valeur. Ces modules de gestion du consentement fonctionnent correctement dans les deux serveurs d'authentification, mais ils ne peuvent être utilisés conjointement (dans les cas d'utilisation de CAS vers l'IdP et de l'IdP vers CAS présentés dans la partie 4.2). En effet, il n'est pas possible de transmettre des consentements de CAS à l'IdP Shibboleth et inversement. Dans ce contexte, il est nécessaire de choisir d'activer la fonctionnalité dans l'un ou l'autre des serveurs en fonction du besoin et ou individuellement pour chaque application.

Pour des usages plus avancés, les mécanismes d'interruption de l'authentification sont adaptables en fonction des besoins en ayant recours par exemple à des scripts Groovy dans le serveur CAS (des exemples ont été exposés dans la présentation collective sur le serveur CAS lors des Esup-Days n°26 [30]). Dans le contexte de l'IdP, il est possible développer de nouveaux flux (*interceptor*).

5.3 Amélioration de la sécurité lors de l'authentification SSO et du SLO

5.3.1 La gestion du cycle de vie des comptes

L'IdP Shibboleth et le serveur CAS n'ont pas vocation à gérer directement les comptes des utilisateurs (leur création, mise à jour, suppression) mais ils sont capables de réaliser des requêtes sur différentes sources d'alimentation. Lors de l'authentification, il peut être intéressant de connaître l'état du compte (actif, suspendu), de vérifier l'application d'une politique de mots de passe…

L'IdP Shibboleth peut gérer par défaut :

- la vérification de l'expiration imminente du mot de passe ; un message est affiché à l'utilisateur avant cette échéance ;
- l'expiration effective du mot de passe ;
- la vérification de l'état d'un compte verrouillé – *locked ;* l'authentification est alors interrompue.

Le serveur CAS offre des possibilités pour aller un peu plus loin que l'IdP Shibboleth en activant le **module *Password Management*.** Par exemple, dans le contexte d'un annuaire LDAP utilisé comme source de données, la LPPE permet de détecter un certain nombre de scénarios et de traiter des problèmes d'authentification liés à :

- l'état du compte (*ACCOUNT_LOCKED, ACCOUNT_DISABLED* et *ACCOUNT_EXPIRED*) ;
- la stratégie de mot de passe (*PASSWORD_MUST_CHANGE* et *PASSWORD_EXPIRED*)

– une authentification non autorisée par rapport à l'heure ou au lieu de connexion (*INVALID_LOGON_HOURS* et *INVALID_WORKSTATION*). L'utilisateur est averti et redirigé le cas échéant vers un système externe de gestion des comptes.

Le module de gestion des mots de passe de CAS offre enfin la possibilité à l'utilisateur de mettre à jour le mot de passe du compte. CAS permet également aux utilisateurs de réinitialiser leurs mots de passe volontairement. Pour cela, l'utilisateur demande sur la page de *Login* de CAS la réinitialisation de son mot de passe. Il reçoit ensuite un lien sécurisé, par mail ou SMS (en fonction de la configuration) valable pendant une fenêtre de temps personnalisable. En suivant ce lien, l'utilisateur peut réinitialiser son mot de passe, après une éventuelle phase de réponse à des questions de sécurité prédéfinies. Une politique de sécurité des mots de passe acceptés peut être configurée dans CAS sous forme d'expression régulière. Une fois que l'utilisateur a modifié son mot de passe, il peut soit être redirigé vers la mire de connexion de CAS pour s'authentifier avec son nouveau mot de passe, soit poursuivre l'authentification automatiquement sur l'application demandée. Cette dernière configuration est recommandée notamment dans le contexte des applications fédérées et dans le contexte de la délégation de l'authentification de l'IdP Shibboleth au serveur CAS.

5.3.2 L'authentification renforcée ou MFA

L'authentification simple (avec un seul facteur d'authentification tel qu'un couple identifiant-mot de passe) peut ne pas être suffisante dans des contextes où la sécurité nécessite d'être renforcée (par exemple pour l'utilisation de fonctions d'administration, l'accès à une application en dehors des horaires de travail, l'accès depuis le réseau externe…). Que ce soit dans l'IdP Shibboleth ou le serveur CAS, il est possible techniquement de mettre en œuvre la demande d'un second facteur d'authentification (OTP, clé U2F, appel à un fournisseur d'authentification renforcée externe…) après le succès du premier.

Le serveur CAS a l'avantage de proposer de nombreux cas d'utilisation du MFA et des intégrations pré-configurées pour différents fournisseurs d'authentification MFA externes. Il est prévu également d'adapter ou rejeter une authentification de l'utilisateur en fonction du contexte d'utilisation (le lieu, le jour ou l'heure, les périphériques de confiance préalablement enregistrés…) et des précédentes authentifications réussies.

L'IdP Shibboleth a l'avantage d'être capable de traiter les demandes d'authentification exprimées sous forme de contraintes par le SP (par exemple avec le profil MFA de REFEDS [19]). Le projet principal de l'IdP Shibboleth ne fournit qu'un seul exemple de configuration du MFA avec le fournisseur externe *Duo Security* . Cependant, des extensions liées au MFA dans l'IdP Shibboleth ont été étudiées par la communauté (TOTP avec *Google Authenticator*, clés U2F…). Nous exposons plus en détails l'authentification MFA dans le contexte de l'IdP Shibboleth, du serveur CAS et des fédérations dans l'article des JRES 2019 [20].

5.3.3 *Logout* et *Single Logout*

La déconnexion d'un serveur d'authentification centralisée et des applications lui déléguant leur authentification est une problématique de sécurité importante. De plus, nous ne pouvons plus nous contenter de recommander à l'utilisateur « de fermer le navigateur utilisé » étant donné que le navigateur peut conserver et recharger de nombreuses informations à la réouverture.

De nombreuses sessions applicatives peuvent être actives en même temps que la session SSO gérée par le serveur d'authentification (IdP Shibboleth ou serveur CAS). Il est important de noter que la session de l'application reste toujours à la charge de cette dernière. La déconnexion (*logout*) et la fin des sessions SSO et applicatives peut se révéler d'une part complexe dans le contexte des fédérations et d'autre part différente en fonction du protocole utilisé (CAS, SAML2, OpenId-Connect…). Par exemple, l'appel au point de terminaison de CAS */cas/logout* n'invalidera que la session SSO de l'utilisateur (dans le contexte de Shibboleth, cet usage du logout n'est pas recommandé).

Une possibilité de déconnexion plus intéressante, nommée SLO (*Single Logout*), va consister à réaliser une déconnexion unique de toutes les applications utilisées. Pour cela, la demande de déconnexion doit

être envoyée aux différentes applications lorsqu'elle est réalisée sur l'une d'elles mais également au serveur d'authentification (toutes les sessions peuvent ainsi être terminées). Pour que le SLO fonctionne, le serveur d'authentification doit être capable de maintenir une liste des différentes applications auxquelles l'utilisateur a accédé. Les applications clientes et le serveur doivent connaître le point de terminaisons correspondant à appeler en cas de déconnexion. La déconnexion SLO ne fonctionne pas dans tous les cas de figure et il est nécessaire de réaliser des tests en fonction des technologies et des stratégies utilisées (*Front-Channel* vs *Back-Channel).* De plus, lorsque l'IdP Shibboleth et le serveur CAS sont utilisés conjointement, il est nécessaire également de déléguer la déconnexion de l'un vers l'autre des serveurs d'authentification en fonction du cas d'utilisation mis en place.

6 Conclusion

Il ressort de cette étude que le serveur d'authentification CAS offre un potentiel plus large de fonctionnalités et qu'il est plus simple à configurer qu'un IdP Shibboleth. En contrepartie, il est plus expérimental (des bugs non identifiés, des fonctionnalités non testées, très peu voire non documentées…). De plus, il est sujet à des évolutions plus rapides (une version majeure par an) qui permettront de profiter de correctifs et de nouvelles fonctionnalités rapidement mais imposeront en contrepartie de mettre à jour et reconfigurer plus souvent le serveur CAS. L'IdP Shibboleth offre un socle principal mieux documenté, plus stable mais avec moins de fonctionnalités. Sa complexité de configuration peut être un frein pour ajouter de nouvelles fonctionnalités, protocoles ou extensions.

À la question (énoncée dans la partie 4) :

> *Puis-je mettre en œuvre une seule brique technique dans mon SI, c'est-à-dire soit un serveur CAS d'Apereo, soit un IdP Shibboleth qui fait tout ?*

La réponse est donc :

Oui, mais en faisant ce choix, il est conseillé de rester le plus proche possible des configurations recommandées par l'une ou l'autre des briques logicielles. En effet, pour des cas bien connus et relativement simples, ces deux serveurs d'authentification sont très proches fonctionnellement et peuvent jouer le rôle de l'un ou de l'autre. Pour les cas d'utilisation et les besoins avancés, il convient de prendre en compte les retours d'expérience des différentes communautés aussi bien françaises qu'internationales et de bien tester la migration avant d'abandonner l'un ou l'autre des serveurs d'authentification. Par exemple, il est nécessaire de vérifier si toutes les applications sont capables de communiquer avec le nouveau serveur d'authentification, notamment celles utilisant des profils spécifiques de la norme SAML 2.0 ou des spécificités ajoutées au protocole CAS.

Il n'est en réalité pas souhaitable de répondre à la question en se basant seulement sur une « vision technologique » des serveurs d'authentification : il est important de prendre en considération des questions liées **aux besoins induits par la gestion des identités et des autorisations** dans l'entité, à **l'organisation interne** (*avez-vous une ou plusieurs équipes différentes pour gérer le LDAP, le serveur CAS, l'IdP Shibboleth, les SP… ? Voulez-vous que ça change ?*), **aux moyens/ressources et compétences disponibles** et aux **« existants monolithiques » du SI** avec lesquels devront s'intégrer les serveurs d'authentification en interne…

Enfin, conformément au cadre technique [31] de la Fédération Éducation-Recherche, l'IdP et le SP Shibboleth sont recommandés car ils bénéficient de documentations et du support technique par l'opérateur de la fédération (RENATER). L'utilisation d'une autre solution logicielle (telles que CAS, LemonLdap ::NG, ADFS, SimpleSAMLphp…) n'est pas proscrite, mais il faut être conscient qu'elle est soumise au **respect du cadre technique et que les tests d'interopérabilité sont à la charge de l'entité qui introduit la solution logicielle**.

Glossaire

ADFS : *Active Directory Federation Services* ;

CAS : *Central Authentication Service* ;

DS : *Discovery Service* ;

FIDO : *Fast IDentity Online* ;

IAM : *Identity and Access Management* ;

IdP : *Identity Provider* - fournisseur d'identité ;

JAAS : *Java Authentication and Authorization Service* ;

JSON : *JavaScript Object Notation* ;

JWT : *Json Web Tokens* ;

LDAP : *Lightweight Directory Access Protocol* ;

LPPE : *Ldap Password Policy Enforcement* ;

MFA : *Multi-Factor Authentication ;*

OTP : *One-Time Password* ;

R&S : *Research and Scholarship* ;

REFEDS : *Research and Education FEDerations* ;

REST : *REpresentational State Transfer* ;

RGPD : *Règlement Général sur la Protection des Données ;*

SAML : *Security Assertion Markup Language* ;

SAML MDUI : *SAML V2.0 Metadata Extensions for Login and Discovery User Interface* ;

SI : Système d'Information ;

SLO : *Single Logout* - déconnexion unique ;

SMS : *Short Message Service* ;

SP : *Service Provider* - fournisseur de service ;

SSO : *Single Sign-On* ;

Supann : Annuaire du supérieur ;

U2F : *Universal Second Factor* ;

URL : *Uniform Resource Locator* ;

TOTP : *Time-based One-Time Password* ;

WAYF : *Where Are You From ?* .

Références

[1] S. Cantor et al., *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS SSTC, Document ID saml-core-2.0-os, http://www.oasis-open.org/committees/security/, mars 2005

[2] *CAS Protocol*, Apereo CAS Documentation, https://apereo.github.io/cas/6.0.x/protocol/CAS-Protocol.html

[3] *Installation d'un SP Shibboleth et Shibbolisation d'application*, Services RENATER, https://services.renater.fr/federation/documentation/guides-installation/sp3/index

[4] *Méta-données SAML publiées par RENATER*, Services RENATER, https://services.renater.fr/federation/documentation/generale/metadata/index

[5] *Metadata Distribution Service Documentation*, Internet2 Wiki, https://spaces.at.internet2.edu/display/MDQ

[6] *Service de découverte de la fédération*, RENATER, https://discovery.renater.fr

[7] *eduPerson*, REFEDS wiki, https://wiki.refeds.org/display/STAN/eduPerson

[8] *Annuaire pour l'enseignement supérieur (supann)*, Services RENATER, https://services.renater.fr/documentation/supann/index

[9] *CAS Architecture*, Apereo CAS Documentation, https://apereo.github.io/cas/6.0.x/planning/Architecture.html

[10] *Catégorie Research & Scholarships*, Services RENATER, https://services.renater.fr/federation/documentation/engagement-conformite/researchandscholarship

[11] *IdP General Architecture*, IdP3 Shibboleth Documentation, https://wiki.shibboleth.net/confluence/display/IDP30/GeneralArchitecture

[12] *Contributions and extensions*, IdP3 Shibboleth Documentation, https://wiki.shibboleth.net/confluence/display/IDP30/Contributions+and+Extensions

[13] *CAS Protocol Configuration*, IdP3 Shibboleth Documentation, https://wiki.shibboleth.net/confluence/display/IDP30/CasProtocolConfiguration

[14] *SAML2 Authentication, Apereo CAS Documentation*, https://apereo.github.io/cas/development/installation/Configuring-SAML2-Authentication.html

[15] *SAML2 Identity Provider Integration w/ InCommon*, Apereo Blog, https://apereo.github.io/2019/01/18/cas61-saml2-idp-incommon/, janvier 2019

[16] *REFEDS MFA Profile with shib-cas-authn3*, Apereo Blog, https://apereo.github.io/2018/02/26/shibcasauthn-duomfa-refeds/, février 2018

[17] *Shib-cas-authn3, A Shibboleth IdP v3.X plugin for authentication via an external CAS Server*, https://github.com/Unicon/shib-cas-authn3

[18] *SAML V2.0 Metadata Extensions for Login and Discovery User Interface Version 1.0*, http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-metadata-ui/v1.0/sstc-saml-metadata-ui-v1.0.html, janvier 2019

[19] *REFEDS Multi-Factor Authentication Profile (MFA)*, https://refeds.org/profile/mfa, février 2018

[20] G. Rousse et L. Auxepaules, *MFA et 2FA dans l'IdP Shibboleth, le serveur CAS d'Apereo et les Fédérations*, JRES 2019, Dijon, décembre 2019

[21] *CASShib*, https://github.com/UniconLabs/casshib

[22] O. Adam et al., *CASShib ou pourquoi j'ai shibbolethisé mon CAS*, JRES 2015, Montpellier, décembre 2015

[23] *Shibbolize a CAS server*, Shibboleth Documentation, https://wiki.shibboleth.net/confluence/display/SHIB2/Shibbolize+a+CAS+server

[24] *Trusted Authentication*, Apereo CAS Documentation, https://apereo.github.io/cas/development/installation/Trusted-Authentication.html

[25] *Pac4j, The Java security engine to protect all your web applications and web services*, http://www.pac4j.org/

[26] *Delegated Authentication to SAML2 Identity Providers*, Apereo Blog, https://apereo.github.io/2019/02/25/cas61-delegate-authn-saml2-idp/, février 2019

[27] *LemonLDAP::NG - Web Single Sign On and Access Management Free Software*, https://lemonldap-ng.org

[28] H. Bougault, V*ers un nouveau modèle de fédération basé sur OpenID Connect ?*, Journées Fédération d'Identités RENATER, FIAP Paris, 25 et 26 septembre 2018

[29] A. Anli et E. Heijligers, *FranceConnect et CAS, ou comment FranceConnect-er vos services numériques à moindre coût*, ESUP-Days n°23 & Apereo Paris 2017, Université Paris Descartes, 31 janvier 2017

[30] D. Lalot et al., *Déploiement de CAS v5: Retours d'expérience de deux établissements*, ESUP-Days n°26, Université Paris Descartes, 15 octobre 2018

[31] *Cadre technique de la fédération Education-Recherche (v1.6.1)*, Services Renater, https://services.renater.fr/federation/rejoindre/cadre-admin-tech/cadre-technique, juillet 2018