2008年度実証実験成果報告 大阪大学サイバーメディアセンター2

大阪大学サイバーメディアセンターに戻る

OpenSSO と Shibboleth 2.0 の SAML 2.0 連携（2008年度実証実験成果報告より）

OpenSSO の紹介

OpenSSO とは Sun Microsystem が提供するオープンソースの Single Sign-On (SSO) 製品で、同社の Sun Java System Access Manager のコードをベースに開発されている。Shibboleth、CAS などと同様に Web SSO の機能を提供する。Sun Java System Access Manager は今後 OpenSSO をベースに製品版の開発も行われる。

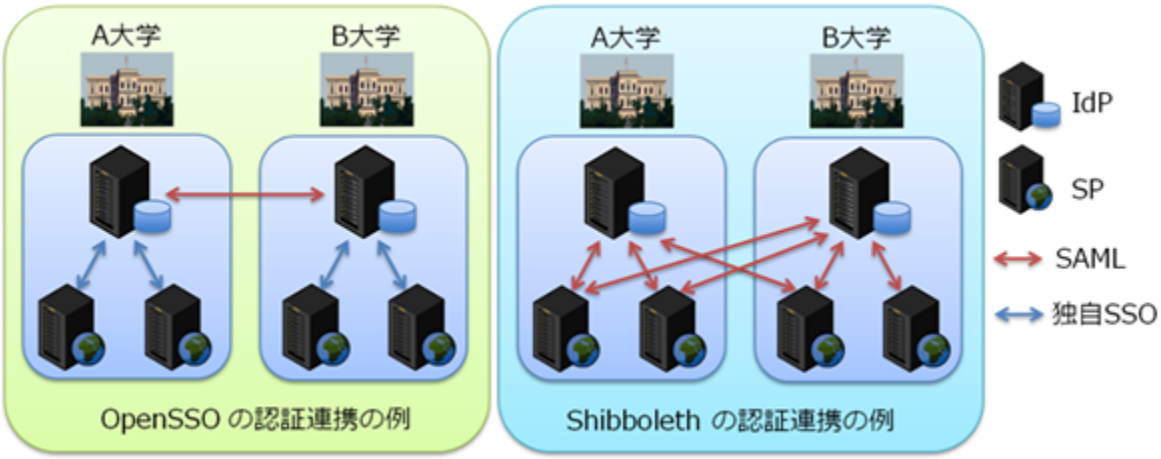
OpenSSO の利点

OpenSSO の利点としては、商用製品をベースに開発が進められているため、Web ベースの IdP 設定ウィザードが用意されていることや、OpenID のような異なる認証方式に対応していること、Web Service の認証に対応していることなどがあげられる。また Sun Microsystems は同社のディレクトリサーバ Sun Java System Directory Server のオープンソース版 OpenDS も開発しており、OpenSSO は OpenDS との相性が良い点も利点であると考えられる。

他に独自の Cookie による SSO も実装しており、簡易な組織内 SSO ではこちらを利用する方法もある。

OpenSSO の欠点

OpenSSO と Shibboleth は IdP、SP に対する想定が異なっている。OpenSSO では「SP はいずれかの IdP の配下にあり、連携は IdP 間で実施」という形が想定しているが、Shibboleth では「SP に対して複数の IdP が連携」という形を想定して実装されている（図 1 参照）。



OpenSSO の連携形態は Liberty Alliance の ID-FF で当初想定されていた連携方式をベースとしている。各 SP を担当する IdP は一意に決定できるため、連携時に連携元の IdP と連携先の IdP の ID を相互に関連づけておくことができれば、SP に対して発生したアクセスを認証する IdP の決定が容易になると考えられていた。特にユーザの操作を介せずに認証処理を行う Web Service では自動的に IdP を決定する機能が求められたため、このような想定が主流となっていた。

- 注) Liberty Alliance の Circle of Trust では Shibboleth のように IdP と SP が直接連携する形態も想定しており、OpenSSO でも Fedlet という SAML2.0 SP の実装が提供されているがまだ発展途上である。

ここで、大規模な大学では IdP と SP が比較的独立した組織、例えば同じ大学でも異なる部局などにおいて運用されるケースが多いため、同一組織内でもシステムの運用レベルが異なり、公開する ユーザ属性を制御する機能などが求められるケースが多い。また、全学で全ての SP が単一の IdP の配下に入るというケースも少なく、認証統合の効果を高めるためにも、IdP と SP が直接個別に連携する形が望ましいケースが多い。このような経緯により Shibboleth では前述のような連携形態を想定している。最終的に連携するサイトが増えるとユーザによる IdP 選択における課題が発生するが、現時点では柔軟性が高い連携方式であると言える。

OpenSSO では組織内の SSO には独自の SSO を用い、簡易に連携できる実装となっている一方で、SAML の機能は後から追加された機能であり、ID マッピングやユーザ属性のアクセス制御の機能がまだ十分に実装されていない。OpenSSO を Shibboleth Federation に参加させるためには、OpenSSO に不足する機能を補うために、後述のようなカスタマイズが必要となる。

なお、利点で述べた OpenSSO の独自 SSO をデフォルトの設定で利用すると、サイト間で同一 Cookie を利用するなどセキュリティ面で脆弱な部分がある。また、SAML2.0 のように SP 単位で厳密なアクセス制御ができないため、以下のブログエントリにあるような問題への対応コストが高くなる。

- ログイン成功時のリダイレクト先として任意サイトの指定が可能であってはいけない(高木浩光@自宅の日記)

上述のように OpenSSO では SAML2.0 SP の実装はそれほど進んでいないため、以下では「OpenSSO と Shibboleth の連携」は基本的に、OpenSSO IdP と Shibboleth SP の連携を指すものとする。

大阪大学において OpenSSO の調査が必要な理由

大阪大学では Sun Java System Access Manager を全学の SSO IdP として採用しており、そのバックエンドとして Sun Java System Directory Server 上でユーザの認証情報を管理している。そのため大阪大学における Shibboleth との連携における選択肢としては、

1. 新規に Shibboleth IdP を構築し、既存の Directory Server を参照する。
2. 既存の Access Manager と Shibboleth SP を SAML2.0 で連携する。

といったアプローチが考えられる。1.については「Shibboleth 2.0 の属性マッピング機能の検証」で述べたように、ある程度柔軟性をもって対応できることが確認できた。しかし、大阪大学の環境では、既存の環境が仮想サーバ等のサーバ追加に対し柔軟性の高い構成をとっておらず、サーバ追加のコストが高かったため、2.の方式についても検討することになった。サーバ追加が容易な場合は 1. のアプローチが望ましいと考えられる。

OpenSSO IdP と Shibboleth SP の連携における注意点

基本的に双方とも SAML2.0 に対応しており、SAML Assertion の扱いに関して互換性がある。しかし、相互に連携するためにはいくつかの対応が必要となる。以下では必要となる対応について述べる。

nameFormat の統一

OpenSSO では SAML Assertion においてデフォルトでは属性の nameFormat が指定されていない。一方 Shibboleth では nameFormat の指定が必須となっており、nameFormat が指定されていない属性は、Shibboleth SP の attribute-map.xml で明示的に unspecified な属性を受理することを記載する必要がある。そのため、OpenSSO と Shibboleth SP で SAML Assertion に記載された属性値を授受する方法は以下の２通りとなる。

1. OpenSSO で uri nameFormat で属性を受け渡す

```
### 以下のパッチの適用が必要
### https://opensso.dev.java.net/issues/show_bug.cgi?id=2775
### build5b 以降は対応済み
### IdP の属性の指定時に uri フォーマットであることを明示的に指定する
### <nameFormat|Assertion 上の属性名=ディレクトリ上の属性名>
urn:oasis:names:tc:SAML:2.0:attrname-format:uri|urn:oid:2.5.4.3=cn
```

2. Shibboleth SP で unspecified nameFormat を受理させる

```
% vi /etc/shibboleth/attribute-map.xml
<Attribute name="urn:oid:2.5.4.3" nameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified" id="cn"/>
```

Federation 内である IdP が unspecified nameFormat での属性提供を許可すると，全てのサイトにおいてその IdP からの属性取得時には unspecified nameFormat を許容する設定が必要になる。そのため運用面では Federation 内の nameFormat をデフォルトの uri フォーマットに統一するのが望ましい。そのため 1. のアプローチをとることにした。

IdP, SP Metadata の違い

基本的に OpenSSO でも Shibboleth SP の Metadata を利用することができるが、<Extensions> タグは Shibboleth の独自拡張であるため、import 時には削除する必要がある。同様に <Organization> や <ContactPerson> などのタグは import 時にエラーになるため、削除する必要がある。

また、Shibboleth SP の attribute-policy.xml のデフォルトの設定では，eduPersonPrincipalName を取得する際に ScopingRules というルールを適用している。このルールでは nobody@example.edu のように、「ユーザ名@ドメイン名」という形式の eppn しか受理しない。<Extensions> を含まない OpenSSO の IdP Metadata をそのまま SP に登録すると、eppn が受理できなくなってしまう。

そのため、

1. IdP Metadata に <Extensions> を追加する

```
<Extensions>
  <shibmd:Scope regexp="false">example.edu</shibmd:Scope>
</Extensions>
```

2. SP の attribute-policy.xml で eppn から ScopingRules 指定をはずす

```
（以下の部分をコメントアウト）
<!--
<afp:AttributeRule attributeID="eppn">
  <afp:PermitValueRuleReference ref="ScopingRules"/>
</afp:AttributeRule>
-->
```

といった対応が必要になる。現状では UPKI 内でも IdP Metadata に Scope 属性を <Extensions> に指定しているサイトは少ないので，2. の対応が妥当であると考えられる。

受け渡し属性選択，変換のためのカスタマイズ

Shibboleth IdP では attribute-resolver.xml における属性値変換、attribute-filter.xml における属性受け渡しポリシーの設定など、SP ごとに自由に受け渡しポリシーを設定することができる。

OpenSSO IdP では、OpenSSO の欠点 で述べたように現時点では柔軟な属性値変換、属性受け渡し可否などのポリシー制御が実施できない。デフォルトでは、ディレクトリ上の属性値をそのまま受け 渡す場合、どの属性を受け渡しするかを IdP につき 1 パターンのみ設定できるようになっている．そのため、属性の変換などが必要な場合はカスタマイズでの対応が必要になる。

OpenSSO IdP では SAML2.0 Assertion を生成における NameID 変換、属性変換を行うクラスのカスタマイズ機能を提供している。以下のクラスをベースに独自の属性変換を行う実装を追加したクラスを実装して jar ファイルを作成し、Tomcat 等の J2EE コンテナのクラスパスに配置する。

```
参照）属性変換を行うデフォルトクラス
com/sun/identity/saml2/plugins/IDPAttributeMapper.java
com/sun/identity/saml2/plugins/DefaultIDPAttributeMapper.java
```

その後管理画面の

「連携」＞「エンティティプロバイダ（IdP）」＞「Assertion Processing」

のメニューで実装したクラスのパスを指定する。

```
例）独自に実装した属性変換クラス
jp.ac.osaka_u.MyIDPAttributeMapper
```

以上のアプローチで OpenSSO においても属性変換が実現できる。

現時点でのカスタマイズ機能にも制限があり、SP ごとの受け渡し属性の制御は実現できていない。提供されている属性変換クラス (IDPAttributeMapper.java) では AuthnRequest（SAML2.0の認証要求）に含まれる SP の entity ID が参照できず、SP ごとに処理を切り替えられないことによる。このような実装になった経緯については NameIDFormat の利用 の節で述べる．

NameIDFormat の利用

SAML2.0 では NameIDFormat を指定することで、IdP，SP 間の ID の対応付けを「一時的な対応付け（transient）」「永続的な対応付け（persistent）」などから選択できる。

```
例）SAML2.0 で指定可能な NameIDFormat の例
urn:oasis:names:tc:SAML:2.0:nameid-format:transient
urn:oasis:names:tc:SAML:2.0:nameid-format:persistent
```

ID の対応付けを選択することで、IdP，SP 間のセキュリティ面の依存関係を選択することができる。

しかし、OpenSSO では NameIDFormat の選択を行うクラス（IDPAccountMapper.java）に SP の情報が受け渡されない実装になっているため、SP ごとに NameIDFormat を指定できなくなっている。該当クラスには SP の entityID ではなく SPNameQualifier が渡されている。これは Google Apps の SAML2.0 SP 実装が AuthnRequest に SPNameQualifier を指定する実装になっているためであるが、Shibboleth SP は AuthnRequest に SPNameQualifier を指定できない。SAML2.0 としては SP の一意な ID としては entityID を指定するのが正しいため、今後Google Apps のような仕様の意図と反する SAML2.0 実装をできる限り減らしていく努力が必要となる。

```
参照）ID変換を行うデフォルトクラス
com/sun/identity/saml2/plugins/IDPAccountMapper.java
com/sun/identity/saml2/plugins/DefaultIDPAccountMapper.java
```

Federation の「エンティティのグループ化」の実装の違い

OpenSSO では、Liberty Alliance に準拠する形でトラストサークル (Circle of Trust) という概念を導入している。認証機能、属性情報の提供は、同じトラストサークルに属する IdP，SP に対して提供する実装になっているので、同じ Federation に参加する IdP, SP のエンティティは、同じトラストサークルに登録する必要がある。

Shibboleth では、IdP おける Metadata 登録時にはこのようなエンティティのグループは意識せずに自 IdP と連携する SP を登録するが、Discovery Service においては、AuthnRequest を送信してきた SP と同じ Metadata に登録されている IdP を候補 IdP としてユーザに提示する。そのため、Discovery Service においてトラストサークル的な概念が実装されていると言える。

これらの違いを認識して運用する必要がある。

Sun Java System Access Manager での UPKI Federation への参加

以上のような対応により OpenSSO IdP と Shibboleth SP の連携が実現できた。このノウハウをベースにして大阪大学の Sun Java System Access Manager も Shibboleth SP と連携することができた。

まとめと今後の課題

OpenSSO での SAML2.0 の利用方法、Shibboleth 2.0 との連携方法を調査し、連携に成功した。また OpenSSO のノウハウを用いて Sun Java System Access Manager と Shibboleth 2.0 の連携にも成功した。

OpenSSO，Sun Java System Access Manager と Shibboleth SP の連携には成功したが、ポリシー制御の面で柔軟性が低く、運用面で課題を残している。一方で、Shibboleth は SAML プロトコルにしか対応していないため、OpenID のような他のプロトコルに対応するためには、別のインスタンスを同時に運用する必要がある。SAML ベースでの厳密な認証の実現と同時に OpenID も含めた軽量なアプリケーションも視野に入れて検討していく上で、CAS も含めてどのシステムがより効率的に運用できるかについて今後引き続き調査が必要である。

- 参照 Fast and Free SSO: A Survey of Open-Source Solutions to Single Sign-On (JavaOne 2007 での Open Source SSO 製品の比較に関する講演)

大阪大学サイバーメディアセンターに戻る