OpenAM 代理認証オプション (mod_authproxy) 管理者ガイド

オープンソース・ソリューション・テクノロジ (株)

更新日　2018 年 8 月 20 日

目次

1 概要

1.1 はじめに

本ドキュメントは、弊社提供の OpenAM 代理認証オプション (Apache モジュール:mod_authproxy、OpenAM の POST 認証プロセスクラス) を導入するための管理者ガイドです。 OpenAM 代理認証オプションのインストールの際に、必ず本ドキュメントの内容を確認してから、作業を実施してください。

1.2 前提事項

mod_authproxy はリバースプロキシ型構成の Apache に導入します。mod_auhtproxy の設定方法については説明しますが、リバースプロキシの proxy 設定や OpenAM は設定済みであること（構築されていること）を前提とします。

OpenAM の POST 認証プロセスクラスは、 OpenAM が稼働する OpenAM サーバーに導入します。インストール手順は、インストールガイドを参照してください。

OpenAM 代理認証オプションを使用することでアプリケーションの Form 認証に対してシングルサインオンを行えますが、 すべての Form 認証に対してシングルサインオンの実現を保証するものではありません。アプリケーションの構成によってはシングルサインオンできない可能性があります。

2 代理認証オプションの説明

本章では、弊社が提供する代理認証オプションの各機能や動作内容について説明します。
以下に、システム全体の概要図を記します。

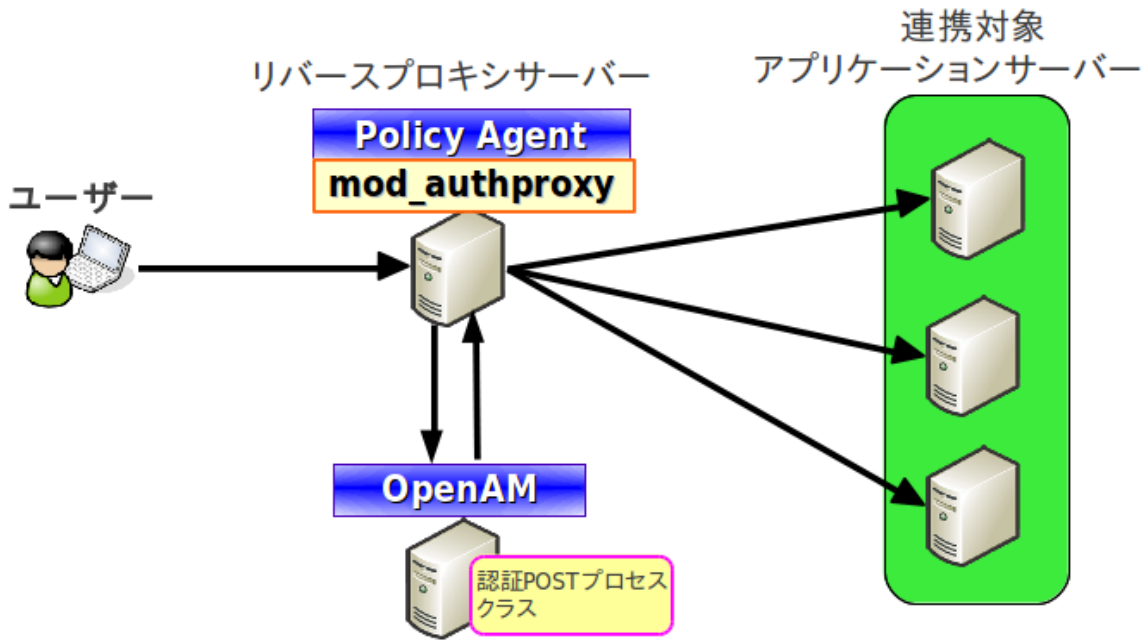

図 1　システム全体概要

代理認証オプションは、リバースプロキシサーバーに導入する「mod_authproxy」、OpenAM サーバーに導入する「認証 POST プロセスクラス」から構成されます。

- リバースプロキシサーバーから OpenAM への通信は、Policy Agent が行います。
- リバースプロキシサーバーとアプリケーションサーバーとの通信は、mod_proxy_http が行います。

2.1 mod_authproxy の機能概要

mod_authproxy は、Apache + OpenAM Apache Policy Agent で構成されるリバースプロキシサーバーに導入することで、リバースプロキシ配下の WEB アプリケーションにシングルサインオン（代理認証）するためのモジュールです。

mod_authproxy は Apache のモジュールです。 OpenAM Agent 経由でアプリケーションの認証に必要な情報（上記図「動作イメージ」内の"認証情報"）を収集し、これをアプリケー

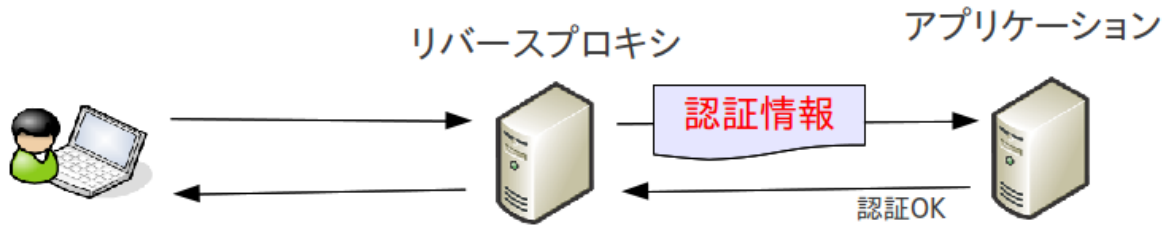

図 2　動作イメージ

ションに送信することで、ユーザーに意識させることなくアプリケーションのログインを完了します。

mod_authproxy を導入することで、次のようなシステムを実現できます。

アプリケーション HTML の Form タグを用いた Form 認証に対して、
OpenAM のログインが完了すればアプリケーションのログインも完了する
シングルサインオンを実現できます。

アプリケーションの BASIC 認証に対して、
OpenAM のログインが完了すればアプリケーションのログインも完了する
シングルサインオンを実現できます。

mod_authproxy には次の機能がありませんので、「Policy Agent」「mod_proxy_http」を併用します。

1. OpenAM への認証済みであることの確認機能
   - 本機能は OpenAM Apache Policy Agent で行いますので、別途 Policy Agent をインストールする必要があります。 Policy Agent は、OpenAM のセッションの確認の実施するとともに、アプリケーションに送信する認証情報を OpenAM から取得します。
2. アプリケーションサーバーに HTTP リクエストを送信 (proxy) する機能
   - 本機能は、Apache の mod_proxy、mod_proxy_http で行います。

2.2 認証ポストプロセスクラスの機能概要

認証ポストプロセスクラスは、OpenAM に導入することで、OpenAM ログイン時にユーザーが入力した「ユーザー名」と「パスワード」、そして BASIC 認証用に「Authorization

ヘッダーの値」を生成し、これらをメモリに保持します。保持した「ユーザー名」、「パスワード」、「Authorization ヘッダーの値」は、OpenAM Agent 経由で mod_authproxy が取得し、 代理認証を行う際に使用されます。

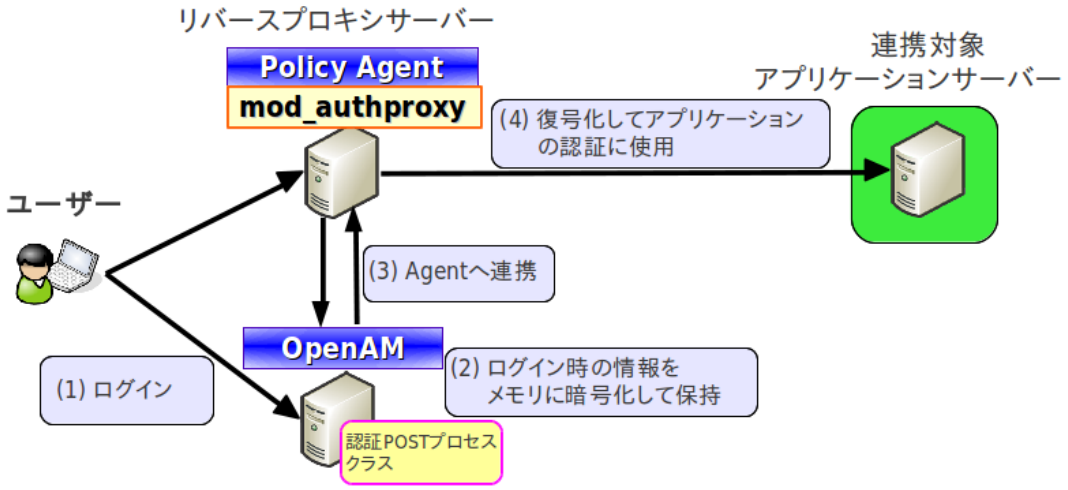

図 3　認証ポストプロセスクラスの機能概要

認証ポストプロセスクラスでは、「パスワード」と「Authorization ヘッダーの値」を暗号化してセッションに保持することができます。暗号化したデータは、mod_authproxy の機能により復号化されます。

この機能により、OpenAM サーバーとリバースプロキシサーバーとの間で、パスワードが暗号化された状態で通信されます。

2.3 mod_authproxy の動作説明

mod_authproxy は、ユーザーとアプリケーション間のリクエストを監視し、アプリケーションからログイン画面が応答されたらアプリケーションに未ログインであると判断し、ユーザーに認証情報を送信させるように動作します。

図 4 のように、mod_authproxy はアプリケーションのログイン画面を認識し、ユーザーが「アプリケーションに未ログインである」ということを判別します。「アプリケーションに未ログインである」と判断すると、アプリケーションに対して代理認証を行います。

代理認証には次の 2 つの方法が用意されており、設定によりどちらかの方法で認証が行われます。

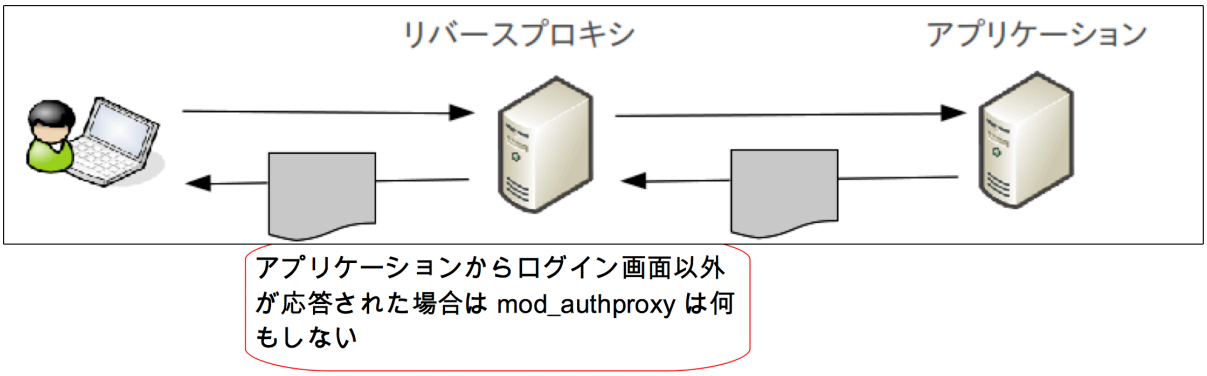

図 4　アプリケーションにログイン済みの (アプリケーションのログイン画面以外が応答された) 場合

1. 自動認証情報送信 HTML による代理認証
2. ダイレクトに認証情報を送信する代理認証

2.3.1 自動認証情報送信 HTML による代理認証

mod_authproxy は、「アプリケーションに未ログインである」と判断すると、アプリケーションのログイン画面の代わりに「自動認証情報送信 HTML（JavaScript で自動的に認証情報を送信する HTML)」を応答し、ユーザーの次のリクエストでアプリケーションの認証が完了するように仕向けます。

送信する認証情報は、OpenAM から取得したデータやログイン画面内に含まれる値を使用できます。

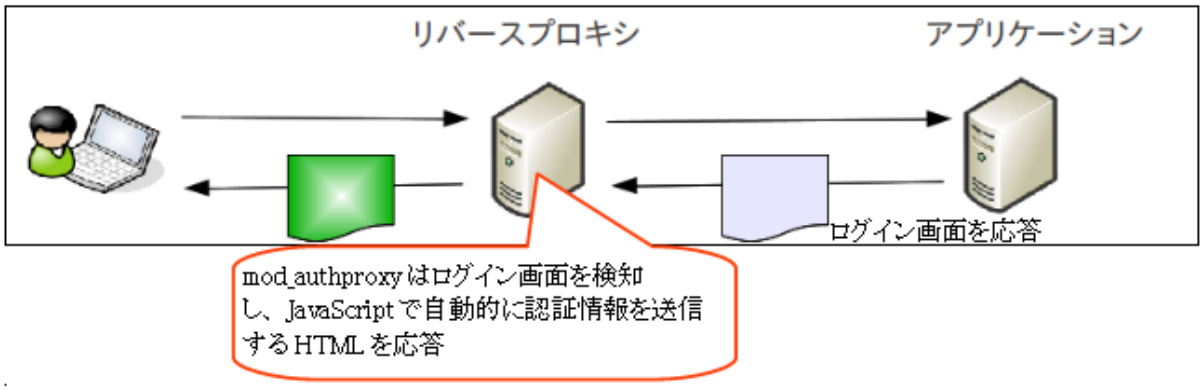

図 5　自動認証情報送信 HTML による代理認証 (アプリケーションがログイン画面応答時)

JavaScript で自動的に認証情報を送信する HTML の例

```
<HTML>
<BODY onLoad='document.authproxy.submit()'>
  <FORM NAME='authproxy' METHOD='POST' ACTION='/login.php'>
    <INPUT TYPE=hidden NAME='username' VALUE='testuser01'>
    <INPUT TYPE=hidden NAME='password' VALUE='password'>
  </FORM>
</BODY>
</HTML>
```

アプリケーションがログイン画面を応答し、mod_authproxy で代理認証を行う際のシーケンスを図 6 に示します。ユーザーは OpenAM に認証済みであり、Policy Agent と OpenAM のやりとりは省略した際のシーケンスです。

2.3.2 ダイレクトに認証情報を送信する代理認証

mod_authproxy は、「アプリケーションに未ログインである」と判断すると、アプリケーションに対して直接ユーザーの認証情報を送信します。送信する認証情報は、OpenAM から取得したデータやログイン画面内に含まれる値を使用できます。ユーザーの 1 つの HTTP リクエストに対して、アプリケーションでは 2 つの HTTP リクエストが届くことになります。(図 7)

2.4 mod_authproxy の提供機能

mod_authproxy が提供する機能を説明します。

- 代理認証機能
  - ログイン画面の判別
  - 自動認証情報送信 HTML の応答
  - 認証情報をダイレクトに送信
- 代理認証のオプション機能
  - 認証データの暗号化/復号化 (ブラウザ – mod_authproxy 間)

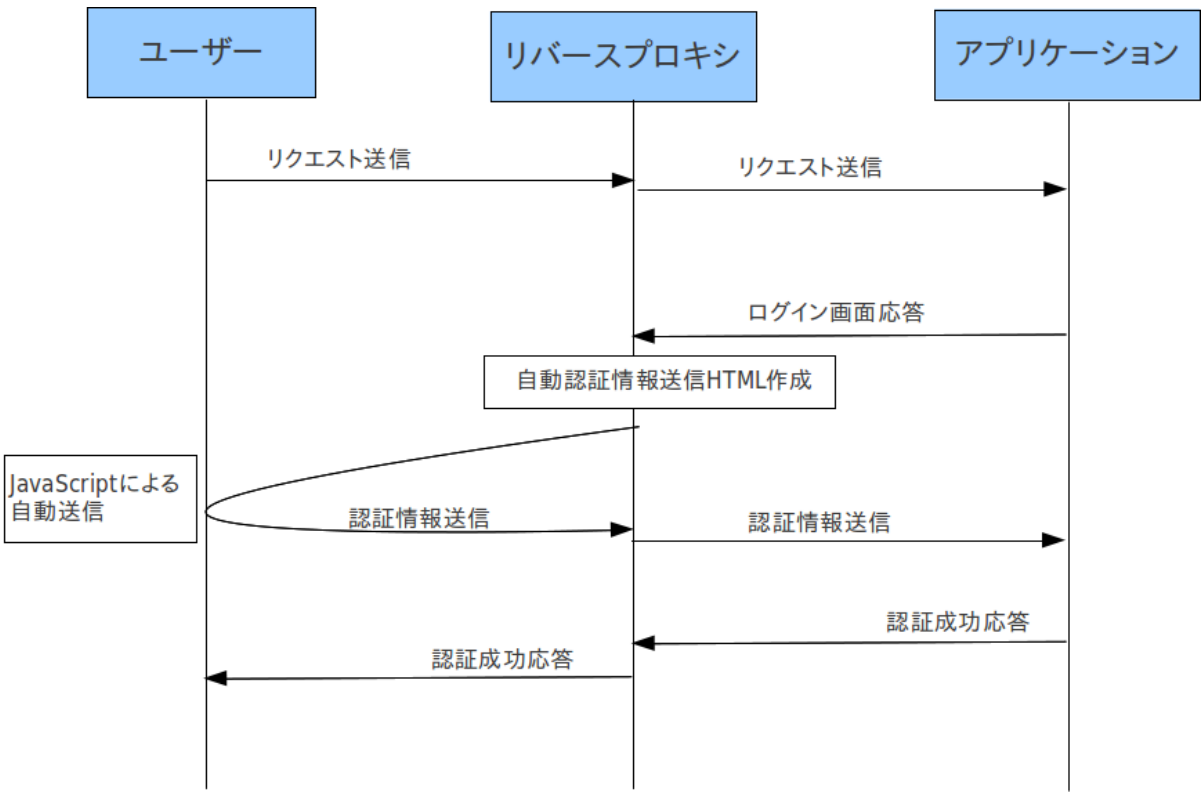

図 6　ログイン画面応答時のシーケンス

- リプレイアタック防止
- ループ防止
- パスワードと Authorization ヘッダー値の復号化

mod_authproxy が提供する機能は上記のみであり、アプリケーションサーバーとの通信 (proxy) は mod_proxy、OpenAM との認証/認可の問い合わせは OpenAM Agent を使用します。

2.4.1 代理認証機能

mod_authproxy は「2.3 mod_authproxy の動作説明」で説明した通り、「ログイン画面の判別」「認証情報の送信（自動認証情報送信 HTML or ダイレクトに送信)」を行います。

この機能を使用するためには mod_authproxy で次の設定を行う必要があります。

- ログイン画面と判断するための条件
- アプリケーションに送信する認証情報

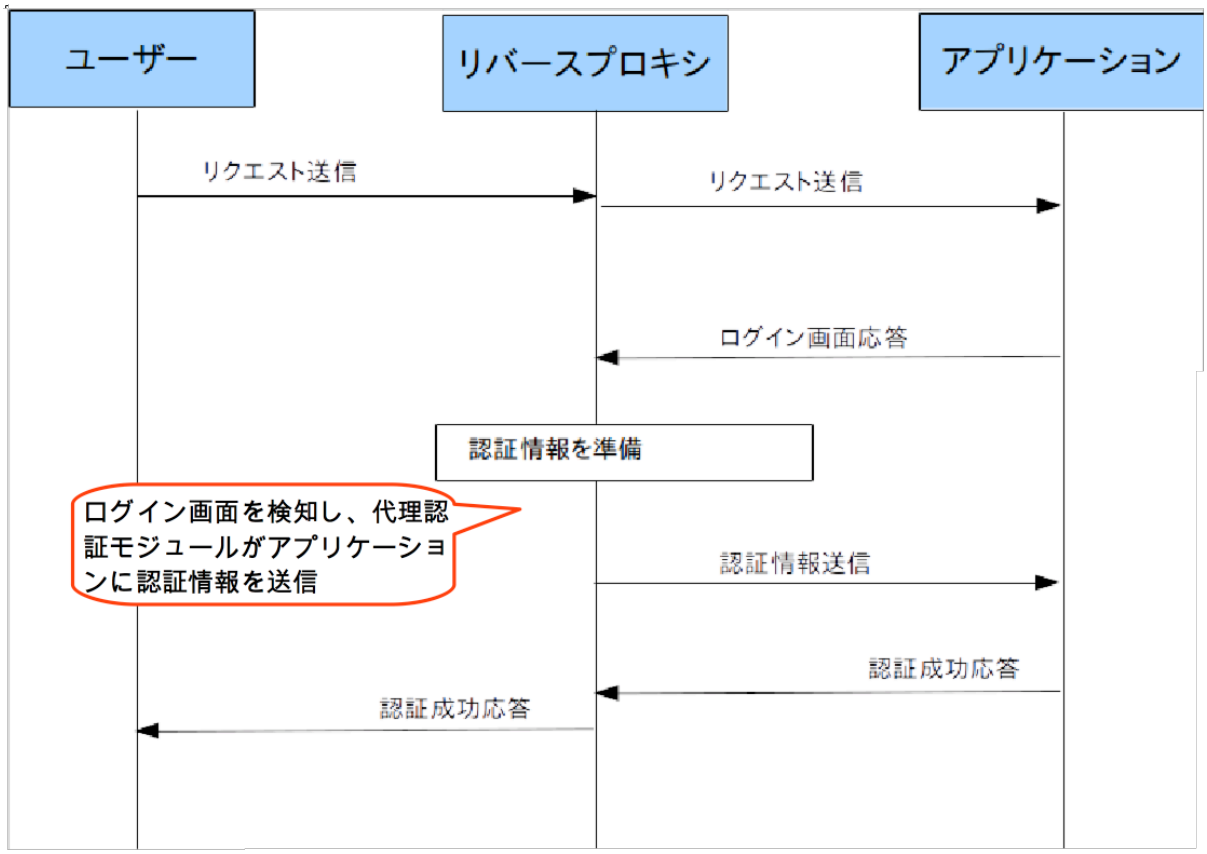

図 7　ダイレクトに認証情報を送信する代理認証

- Form タグ名や認証先 URL の設定（必要に応じて）

設定方法は、「4 mod_authproxy の導入」にて記載します。

2.4.2 代理認証オプション機能

代理認証のオプション機能を説明します。

2.4.2.1 認証データの暗号化/復号化 (自動認証情報送信 HTML 利用時)

代理認証モジュールを使用すると、アプリケーションに送信する認証情報がブラウザを経由するため、ユーザーが認証情報を参照することが可能です。

通常この認証情報はユーザーが知りうる情報のため、参照できることにセキュリティのリスク等の問題はありません。しかし、それでもユーザーに参照できないようにさせたい場合に自動認証情報送信 HTML に含まれる認証情報を暗号化することが出来ます。

暗号化された情報は、次のリクエストでリバースプロキシサーバーに送信された際に復号化されます。

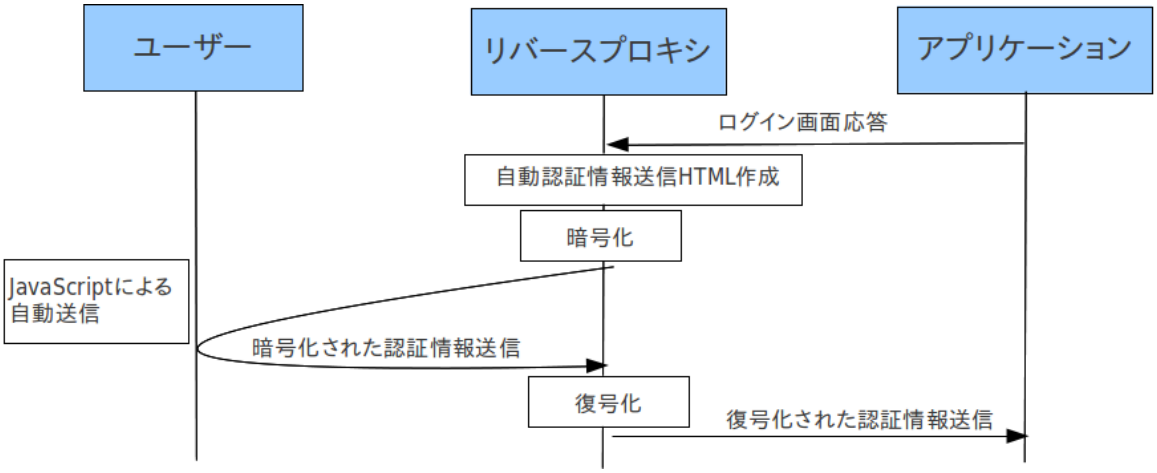

図 8　暗号・復号処理時のシーケンス

2.4.2.2 リプレイアタック防止 (自動認証情報送信 HTML 利用時)

mod_authproxy を使用すると、アプリケーションに送信する認証情報がブラウザを経由するため、ユーザーが認証情報を保持しておくことが可能です。後日その保持した情報をアプリケーションに送信するとログイン出来てしまう場合があります。これを防止するため、mod_authproxy が生成した認証情報送信データに有効期限 (タイムアウト) を定義できます。

タイムアウトは秒単位で設定します。なおリプレイアタック防止機能を使用するためには、認証データの暗号化/復号化機能を併用する必要があります。

2.4.2.3 ループ防止 (自動認証情報送信 HTML 利用時)

代理認証動作時に何らかの理由によりアプリケーションサーバーのログインが完了せず、常にログイン画面が返ってしまう事態が発生すると、認証のループが発生します。

このループを防止するため、mod_authproxy は「自動認証情報送信 HTML」応答時にループ防止の Cookie を発行します。mod_authproxy の設定では Cookie 名を任意の名称に変更できます。

リバースプロキシサーバーを経由しない代理認証の場合は、本機能を無効にする必要があります。

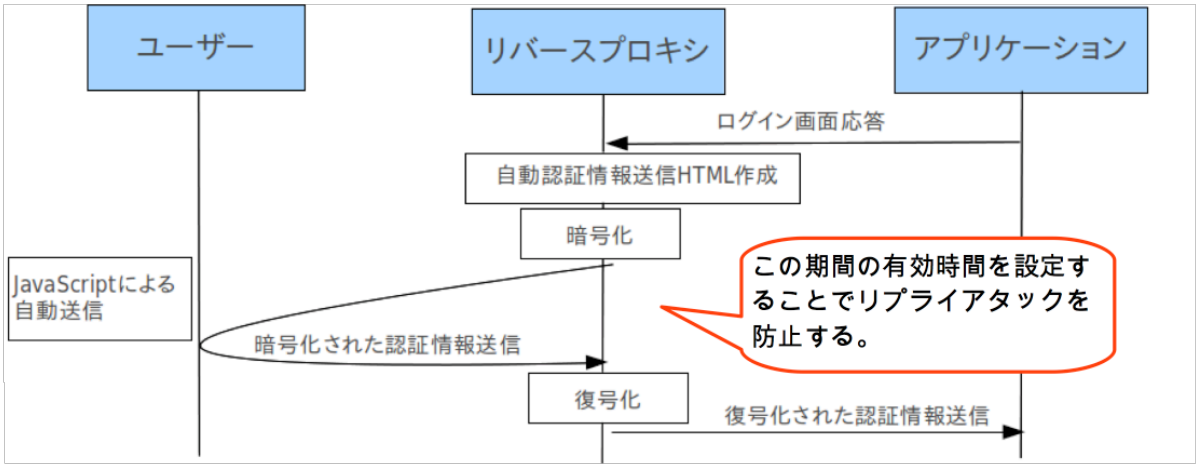

図 9　リプレイアタック防止

2.4.2.4 パスワードと Authorization ヘッダー値の復号化

OpenAM にて「OpenAM ポスト認証プラグイン」を導入すると、ユーザーが入力したパスワードと Basic 認証で利用する Authorization ヘッダーの値を暗号化することができます。mod_authproxy では、この暗号化されたデータを復号し利用することができます。

この機能は「OpenAM ポスト認証プラグイン」と組み合わせて使用します。本機能により、OpenAM サーバー – リバースプロキシサーバー間ではパスワードが暗号された状態でやりとりが行われます。

3 認証ポストプロセスクラスの導入

本章では、認証ポストプロセスクラスインストール後の設定方法について説明します。

3.1 認証ポストプロセスクラスの読み込み

OpenAM に認証ポストプロセスクラスを読み込む設定を行います。作業は OpenAM 管理コンソールより行います。

1. OpenAM に管理者 (amAdmin) でログインします。
2. OpenAM 管理コンソールで認証ポストプロセスクラスを設定する画面を表示します。
   - OpenAM 9.x/11.x
     - 「アクセス制御」 → 「[レルム]」 → 「認証」 →「すべてのコア設定」をクリック
   - OpenAM 13.x
     - 「レルム」→「[レルム]」→「認証」→「設定」→「ポスト認証プロセス」をクリック
3. 認証ポストプロセスクラスにて「jp.co.osstech.oam.authentication.pap.AuthProxyPAP」を登録します。
4. 画面右上の「保存」をクリックします。

以上で完了です。

3.2 認証 POST プロセスクラスのプロパティファイルの設定

認証 POST プロセスクラスのプロパティファイルを設定します。war 名が openam の場合、プロパティファイルの配置場所は下記のとおりです。

設定が完了したら tomcat の再起動を行い、設定内容を反映させます。

OpenAM 9.x の場合

```
/opt/osstech/share/tomcat6/webapps/openam/WEB-INF/classes/AuthProxyPAP.properties
```

OpenAM 11.x/13.x の場合

```
/opt/osstech/share/tomcat7/webapps/openam/WEB-INF/classes/AuthProxyPAP.properties
```

プロパティファイルで設定する項目は下記のとおりです。

項目名	初期値	説明
AESPassword	AuthProxy	パスワード、Authorization ヘッダー値を暗号化する際の鍵となる文字列を設定します。
EncryptPassword	true	ユーザーが OpenAM ログイン時に入力したパスワードを保持する際の暗号化有無を設定します。 true・・・暗号化して保持します。 false・・・暗号化しません。
EncryptAuthorizationData	true	OpenAM がセッション値としてセットする Authorization ヘッダー値の暗号化有無を設定します。 true・・・暗号化して保持します。 false・・・暗号化しません。
BasicAuthXX	なし	下記「※ BasicAuthXX について」を参照ください。

【※ BasicAuthXX について{#BasicAuthXX について}】

本項目では、次の三つを定義します。

- 「OpenAM のセッション値としてセットする名前」
- 「ユーザー ID が格納されている LDAP 属性名」
- 「パスワードが格納されている LDAP 属性名 または　固定文字列」

設定では、カンマ区切りにて次の順序で定義を行います。固定文字列のときはダブルクォーテーションで囲みます。

```
"セッション値の名前", "ユーザー ID の LDAP 属性名", "パスワードの LDAP 属性名（or 固定文字列)"
```

本設定項目は、”BasicAuth” の後に任意の数字を入れることで、複数定義することが可能です。

「セッション値の名前」は

- AuthorizationData01
- AuthorizationData02

とし、以降後ろの数字を変えたネーミングルールとします。(「セッション値の名前」は

Agent の設定の「セッション属性のマップキー」と合わせればどんな名称でも良いですが、ユーザーが意識しない内部の名称となるため、AuthorizationData[数値] というルールとします。）

設定例：

```
BasicAuth01=AuthorizationData01,sn,Password
BasicAuth02=AuthorizationData02,mail,Description
BasicAuth03=AuthorizationData03,uid,"password"
```

OpenAM に入力した「ユーザー ID」と「パスワード」を BASIC 認証に使用する場合は、認証ポストプロセスクラスのデフォルト動作でセッション名「AuthorizationData」としてセットされるため、本項目で定義する必要はありません。逆に言えば、本項目は「OpenAM に入力したユーザー ID とパスワード」"以外の"値を、BASIC 認証のシングルサインオンとして利用する場合に設定します。

3.3 Policy Agent のセッション設定

認証ポストプロセスクラスでセットした OpenAM セッションの情報を mod_authproxy で使用出来るようにするため、Agent の設定にて HTTP ヘッダーにセットします。

作業は OpenAM 管理コンソールより行います。

1. OpenAM に管理者 (amAdmin) でログインします。
2. OpenAM 管理コンソールで Agent の「セッション属性処理」を設定する画面を表示します。
   - OpenAM 9.x/11.x
     - 「アクセス制御」→「[レルム]」→「エージェント」→「web」→「[対象のエージェント]」→「アプリケーション」をクリック
   - OpenAM 13.x
     - 「レルム」→「[レルム]」→「エージェント」→「web」→「[対象のエージェント]」→「アプリケーション」をクリック
3. 「セッション属性処理」の「セッション属性フェッチモード」を”HTTP_HEADER” にチェックを入れます。
4. 「セッション属性マップ」に次の値を設定します。

マップキー	対応するマップ値
UserToken	(任意の値) ※ 1

マップキー	対応するマップ値
Password	(任意の値) ※ 1
AuthorizationData または BasicAuthXX の” セッション値の名前” ※ 2	Authorization ※ 3

【※ 1】 ここで設定した値が、AuthProxyLoginData の %{ヘッダー名}i で設定する「ヘッダー名」となります。

【※ 2】アプリケーションと BASIC 認証のシングルサインオンを行う場合に必要です。BASIC 認証のシングルサインオンを行わない場合、設定は不要です。AuthorizationData と指定すると、「OpenAM に入力したユーザー ID とパスワード」が Authorization ヘッダーにセットされます。

【※ 3】 ここでは Authorization ヘッダーに直接セットしていますが、アプリケーション毎に BASIC 認証のシングルサインオンを行う場合は、 Policy Agent の設定では一時的に別の HTTP ヘッダー名にセットし、mod_headers で Authorization ヘッダーにセットさせる定義を行う必要があります。「7.2.15 AuthProxyAuthorizationName ディレクティブ」の設定例を参考にしてください。

5. 画面右上の「保存」を押します。
6. アプリケーションに HTTP ヘッダーでパスワードが送信されることを防ぐため、リバースプロキシサーバーの httpd.conf に以下の設定を行います。
   - `RequestHeader unset [Password` **に対応するマップ値に設定した値**`]`

以上で、作業は完了です。

4 mod_authproxy の導入

本章では、mod_authproxy インストール後の設定方法について説明します。設定にあたっては、mod_authproxy の動作やシングルサインオンを行う先のアプリケーションのログインについても理解する必要があります。

なお、本手順はアプリケーションの認証が Form 認証の場合に必要な作業です。アプリケーションの認証が BASIC 認証の場合は、 4.3.4 パスワードと Authorization ヘッダー値の復号化機能を実施することでシングルサインオンを実現できます。その他の作業は不要です。

4.1 アプリケーションのログイン情報の収集

mod_authproxy の設定を行うため、認証連携先のアプリケーションの情報を収集します。mod_authproxy では「ログイン画面と判断するための条件」「アプリケーションに送信する認証情報」を設定するため、これらの情報を収集します。

4.1.1 ログイン画面と判断する条件

mod_authproxy では、次の条件を満たしたコンテンツをログイン画面と判断します。

- ログイン画面を表示するための URL とマッチするか
- ログイン画面の Content-Type が text/html であるか
- ログイン画面内に含まれる文字列が設定値とマッチするか

これらの設定を行うために、各情報を収集します。

4.1.1.1 ログイン画面を表示するための URL

アプリケーションで「この URL にアクセスするとログイン画面が表示される」という URL を調査します。

通常のアプリケーションでは「/login.php」のようにログイン画面表示用の URL が用意されていると思われますので、その URL を調べます。アプリケーションによっては、URL に関わらず、未ログイン状態 (セッションが無い、セッションが有効でない) であるときにログイン画面が表示される、という場合があります。この場合は、ログイン画面が表示される可能性のある URL、例えば「/配下」または「/[war 名]/配下」等の○○○配下を調査します。

4.1.1.2 ログイン画面の Content-Type

ログイン画面の Content-Type としては、text/html に対応します。ログイン画面の Content-type が text/html 以外である場合の動作は保証されません。

4.1.1.3 ログイン画面内に含まれる文字列

アプリケーションの「ログイン画面だけ」に含まれる文字列を調査します。

ログイン画面以外のコンテンツ応答時、mod_authproxy が動作しないようにするために設定します。title 等、他の画面では使われておらず、ログイン画面だけに含まれている文字列を調査します。

ログイン画面だけに含まれている文字列を見つけるのが困難な場合、ログイン画面を編集し、任意の文字列を埋め込みます。下記のように、html のコメントにてログイン画面だけに任意の文字列を含める方法が、調査も不要となり確実です。

```
<!-- 8fb6f57c3e3993407d48e985e423a72c6dba36fe -->
```

なお、mod_authproxy では「ログイン画面に含まれない文字列」の指定も可能です。この場合、指定した文字列が含まれていないことが、ログイン画面であると判断する条件となります。

4.1.2 アプリケーションに送信する認証情報

mod_authproxy に、アプリケーションに対して送信する認証情報を設定します。設定するために「アプリケーションに送信する認証情報は何か」や「どこから情報を取れば良いか」を調査する必要があります。調査手順を示します。

4.1.2.1 事前準備

アプリケーションにログインする環境を準備します。（リバースプロキシ経由、アプリケーションサーバー直接のどちらのシステム構成でも構いません。）

4.1.2.2 HTTP リクエストのキャプチャー取得

ブラウザの開発者用ツール等を用い、アプリケーションのログイン画面にて認証に必要な情報を入力して、ログインボタン押下時の POST データを取得します。

下記に、実際のサンプルとして、あるアプリケーション (xoops) を利用した際に取得した POST データを示します。

```
uname=user01&pass=Password%21&xoops_redirect=%2Fhtml%2F&op=login&submit=Login
```

4.1.2.3 アプリケーションのログイン画面の取得

ブラウザでアプリケーションのログイン画面を表示し、HTML のソースを表示します。ログイン時に送信する Form タグを確認します。

下記に、実際のサンプルとして、あるアプリケーション (xoops) のログイン画面の Form タグを示します。

```
<form action="/html/user.php" method="post" style="margin-top: 0px;">
    <br />
    Username:<br />
    <input name="uname" id="legacy_xoopsform_block_uname"
        type="text" size="14" maxlength="25" value="" />
    Password:<br />
    <input name="pass" id="legacy_xoopsform_block_pass"
        type="password" size="14" maxlength="32" />
    <br />
    <input type="hidden" name="xoops_redirect" value="/html/" />
    <input name="op" id="legacy_xoopsform_block_op"
         type="hidden" value="login" />
    <br /><br />
    <input name="submit" id="legacy_xoopsform_block_submit"
         type="submit" value="Login" />
</form>
```

取得した POST データとログイン画面から、アプリケーションに送信する認証情報の「name」「value」「値の発生元」を調べます。調査結果を下記表のようにまとめるとわかりやすくなります。

POST データの name	POST データの value	値の発生元
uname	user01(ユーザー ID)	ユーザーの入力
pass	Password!(パスワード)	ユーザーの入力
xoops_redirect	/html/	ログイン画面内 input タグ内
op	login	ログイン画面内 input タグ内
submit	Login	ログインボタン

このように、アプリケーションに送信すべき認証情報が表の通りとわかります。この情報

を送信するように仕向ける HTML を、mod_authproxy で作成することになります。

ここで値の発生元について説明します。mod_authproxy で用意できる送信情報は、次の通りです。

- リクエスト HTTP ヘッダーの値（OpenAM Agent から受け取った値）
- Form タグの input タグ内の値
- 設定ファイルに記載した固定の値

調査した認証情報が、mod_authproxy で用意できるかを確認します。特にユーザーが入力する値に関しては、リクエスト HTTP ヘッダーから取得できる値かどうかが重要となります。

OpenAM Agent では、OpenAM のセッション情報やデータストアの値をリクエスト HTTP ヘッダーに付与できるため、OpenAM が用意できる値であれば用意できます。調査した認証情報において、OpenAM を通して用意できない値が一つでもあった場合、代理認証を行うことはできません。

4.1.3 Form タグ名や送信先 URL の情報

mod_authproxy では、アプリケーションのログイン画面から input タグの属性値や認証情報送信先 URL(Form タグの action 属性) を取得するため、ログイン画面内の Form タグを精査します。

ログイン画面内に Form タグが複数ある場合は、authProxy の設定にて Form 名を指定する必要があります。ログイン画面内に「Form タグが複数ないか」「複数ある場合は、ログインに使用する Form タグの Form 名は何か」を調査します。

なお、ログイン画面によっては「Form タグが複数ある」が「Form 名の記述がない」場合があります。mod_authproxy の設定にて Form 名を指定しない場合は、下記の動作となります。

- コンテンツ内の全ての Form タグ input タグ内の属性名、属性値を保持する。
- 属性名が重なった場合は後に見つかった属性値が優先される。
- Form タグ action 属性も後に見つかった属性値が優先される。

action 属性は、mod_authproxy に設定されていれば、その設定値が優先されます。このため、Form タグが複数ありかつ Form 名がない場合でも、action 値 (送信先 URL) を予め設定内に含めておけば、任意の URL に認証データを送信できます。

4.2 mod_authproxy の設定

アプリケーションの調査を終えたら、実際に mod_authproxy の設定を行います。mod_authproxy は Apache のモジュールであるため、httpd.conf などの Apache の設定ファイルにて設定を行います。

インストールを行うと、/etc/httpd/conf.d/authproxy.conf にサンプル conf ファイルが配置されています。標準構成では、Include ディレクティブで/etc/httpd/conf.d/配下が自動的に読み込まれますので、この authproxy.conf に設定することもできます。

4.2.1 Location ディレクティブ

設定のベースとなる Location ディレクティブを定義します。

Location ディレクティブの URL には、アプリケーションのログイン画面で表示する URL です。「4.1.1 ログイン画面と判断する条件」で調査した URL を設定します。

設定例：

```
<Location /login.php>
    ～［ここに以降の定義を記載］～
</Location>
```

以降の 4.2 の定義は、特に明示しない限り、すべてディレクティブに囲まれた中に記載します。

4.2.2 output フィルターの登録

mod_authproxy は、Apache の output フィルターとして動作します。このため、mod_filter を使って mod_authproxy を動作させる設定を行います。mod_authproxy のフィルタ名は「authproxy」です。

Content-Type が text/html のときに動作する設定を行います。Apache2.2 と Apache2.4 で設定方法が異なります。

設定例 (Apache2.2 の場合)：

```
FilterProvider authproxy  authproxy  resp=content-type $text/html
FilterProtocol authproxy  "change=yes"
FilterChain authproxy
```

設定例 (Apache2.4 の場合)：

```
FilterProvider authproxy  authproxy "%{CONTENT_TYPE} =~ m|^text/html|"
FilterProtocol authproxy  "change=yes"
FilterChain authproxy
```

なお、Apache2.2 の場合はデフォルトで mod_filter の LoadModule ディレクティブがコメントアウトされているので、必要に応じ、コメントアウトを外して有効にします。

リバースプロキシ構成の場合、URL 変換等で他のフィルタを使用する場合を考慮する場合があります。例えば、文字列の変換を行う mod_substitute と組み合わせる場合は、下記のようになります。

設定例 (Apache2.2 の場合)：

```
FilterProvider authproxy  authproxy  resp=content-type $text/html
FilterProvider SUBSTITUTE SUBSTITUTE resp=content-type $text/html
FilterProvider SUBSTITUTE SUBSTITUTE resp=content-type $text/plain
FilterProtocol authproxy  "change=yes"
FilterProtocol SUBSTITUTE "change=yes"
FilterChain authproxy SUBSTITUTE
```

4.2.3 ログイン画面に含まれる文字列の登録

次に、アプリケーションのログイン画面に含まれる文字列を登録します。文字列の登録は、複数の値を設定できます。

「4.1.1 ログイン画面と判断する条件」の例の通りに設定する場合は、ログイン画面に追加した任意のコメントを設定します。

```
AuthProxyLoginKeyword “<!-- 8fb6f57c3e3993407d48e985e423a72c6dba36fe -->”
```

4.2.3.1 ユーザー ID とパスワード

ユーザー ID とパスワードは、OpenAM でログインした値を使用します。

弊社で構築した OpenAM（OpenAM ポスト認証プラグイン導入）の場合は、図 10 に例示するように、OpenAM ログイン時の ID とパスワードを OpenAM Agent で HTTP ヘッダーに追加できます。図 10 の設定例では、HTTP リクエストヘッダー対し、「userid」という名称でアクセス者のユーザー ID を、「authpass」という名称でアクセス者のパスワードを格納しています。

なお、このままでは、リバースプロキシ配下のアプリケーションにユーザーのパスワードが HTTP ヘッダーの形態で送信されてしまいます。このため、HTTP ヘッダーとしてはパス

図 10　アプリケーションに送信する認証情報

ワードを送信しないようにするための設定を、＜ Location ＞の範囲外に設定します。

```
RequestHeader unset authpass
```

リクエスト HTTP ヘッダーにユーザー ID、パスワードが入っていますので、それぞれ設定は以下の通りとなります。

```
AuthProxyLoginData uname %{userid}i
AuthProxyLoginData pass %{authpass}i
```

この設定により、アプリケーションに送信される認証情報として、「uname」という名前で値は「ログインしたユーザー ID」、「pass」という名前で値は「OpenAM にログインした際のパスワード」が使用されます。

次に、ログイン画面内の input タグの値を設定します。

```
AuthProxyLoginData xoops_redirect %{xoops_redirect}n
AuthProxyLoginData op %{op}n
```

この設定により、mod_authproxy はアプリケーションのログイン画面の input タグから値を取得します。

「xoops_redirect」という名前で値には「input タグの名前 xoops_redirect の値」、「op」という名前で値には「input タグの名前 op の値」が使用されます。

4.2.4 Form 名と送信先 URL

ログイン画面に複数の Form タグが存在する場合、Form 名の定義、または、送信先の URL の定義を行います。Form タグが一つしかない場合は設定する必要はありません。例えば下記のように Form タグが二つある場合、ログインで使用する form タグの名称を指定します。

```
<form action="/html/usera.php" name="loginA" method="post" ><br />
Username:<br />
<input name="uname" type="text" size="14" maxlength="25" value="" />
    Password:<br />
    <input name="pass" type="password" size="14" maxlength="32" /><br />
<input type="hidden" name="xoops_redirect" value="/html/" />
<br /><br />
<input name="submit" type="submit" value="Login" />
</form>

<form action="/html/userb.php" name="loginB" method="post" ><br />
Username:<br />
<input name="uname" type="text" size="14" maxlength="25" value="" />
    Password:<br />
    <input name="pass" type="password" size="14" maxlength="32" /><br />
<input type="hidden" name="xoops_redirect" value="/html/" />
<br /><br />
<input name="submit" type="submit" value="Login" />
</form>
```

アプリケーションでは loginA でログインを行うとすると、次のように設定します。

```
AuthProxyFormName loginA
```

"Form タグが二つ存在し"かつ「Form 名がない」というような場合等では、認証情報送信先 URL を action 属性の値ではなく設定した URL にするため、下記のように設定します。

```
AuthProxySendUrl /html/user.php
```

4.2.5 まとめ

これまで設定した内容を一つにまとめると、下記の通りとなります。下記例は Apache2.2 の場合の設定方法です。 前述したとおり、Apache2.4 の場合は mod_filter のコンテントタイプの設定方法が異なります。

```
RequestHeader unset authpass
<Location /login.php>
    FilterProvider authproxy  authproxy  resp=content-type $text/html
    FilterProtocol authproxy  "change=yes"
    FilterChain authproxy
    AuthProxyLoginKeyword “<!-- 8fb6f57c3e3993407d48e985e423a72c6dba36fe -->”
    AuthProxyLoginData uname %{userid}i
    AuthProxyLoginData pass %{authpass}i
    AuthProxyLoginData xoops_redirect %{xoops_redirect}n
    AuthProxyLoginData op %{op}n
    AuthProxyFormName loginA
</Location>
```

この例のシステムでは、 http://[FQDN]/login.php にアクセスすると mod_authproxy が動作し、アプリケーションの認証が自動的に完了します。

複数のアプリケーションサーバーに対してそれぞれ代理認証を行う場合、この＜ Location ＞で囲まれた構成をアプリケーションサーバーの数だけ定義することになります。

以上で、代理認証の設定が完了です。

4.3 mod_authproxy のオプション設定

「4.2 mod_authproxy」の設定に従い導入すれば代理認証は行えます。ここでは、必要に応じて行う代理認証のオプション設定について説明します。

4.3.1 認証データの暗号化/復号化機能

認証データの暗号化/復号化機能を利用するための設定方法を説明します。

まずはじめに、暗号化の鍵文字列の定義を行います。設定値は任意の文字列でかまいませんが、リバースプロキシを冗長化のため複数台構成にしている場合は、すべてのサーバーで同じ鍵文字列を設定します。

鍵文字列の設定は Apache のサーバー単位で行うため、＜ Location ＞に囲まれた範囲には記述しません。

```
AuthProxyEncryptPassword 1234567890
```

次に、暗号化を行うデータを選択します。今回はユーザー ID とパスワードを暗号化することにします。その場合は、下記のとおり AuthProxyLoginData の第 3 引数に”encrypt” と入力します。

```
AuthProxyLoginData uname %{userid}i encrypt
AuthProxyLoginData pass %{authpass}i encrypt
```

最後に、送信されてきたデータの復号化を行う設定を行います。

復号は Apache の input フィルターで行います。input フィルターはログイン情報送信先 URL へのアクセス時に動作させるため、Form タグ内の aciton 属性値や AuthProxySendUrl ディレクティブで定義した URL を＜ Location ＞で指定します。input フィルター名は「authproxy」です。

```
<Location /html/user.php>
    setInputfilter authproxy
</Location>
```

なお、一つの value 値あたりに暗号化可能なデータサイズには 4096byte の上限があります。Value 値のサイズが 4096byte を越えると暗号化されません。

4.3.2 リプレイアタック防止機能

リプレイアタック防止機能を利用するための設定について説明します。

リプレイアタック防止機能を利用するためには、認証データの暗号化/復号化処理を使用していることが前提となります。そのため「4.3.1 認証データの暗号化/復号化機能」に従い、何か一つでも認証データの暗号化/復号化を行う必要があります。

次に、リプレイアタック防止機能で許容する有効期間を定義します。この期間を過ぎたアクセスに対しては、Apache のエラーが応答されます。デフォルトの有効期間は 60 秒です。下記に 3 分 (180 秒) を設定する例を示します。

なお、有効期間の設定は Apache のサーバー単位で行うため、＜ Location ＞に囲まれた範囲には記述しません。

```
AuthProxyLoginDataTTL 180
```

デフォルト値が 60 秒のため、認証データの暗号化/復号化を行うとリプレイアタック防止

機能が有効となります。リプレイアタック防止機能を使用したくない場合は、次のように 0 を指定します。

```
AuthProxyLoginDataTTL 0
```

なおエラー画面の応答は、使用する Apache ハンドラによります。(Apache2.2.3 の mod_proxy_html の場合は 400 エラーとなることを確認しています。)

リプレイアタック防止機能によりアクセスが有効期限外となった際、下記のエラーがエラーログに出力されます。

```
[error] [client IP アドレス] authProxy input filter Login Data Send
 Timeout:[経過時間 (秒)] User:[ユーザー名]
```

4.3.3 ループ防止機能

ループ防止機能は標準で有効にされている機能であり、Cookie を利用します。Cookie 名のデフォルトは「authproxy」です。例として Cookie 名を osstechauth に変更する場合は、下記の通り設定します。

なお、Cookie 名の指定は Apache のサーバー単位で行うため、< Location >に囲まれた範囲には記述しません。

```
AuthProxyCookieName osstechauth
```

リバースプロキシを経由しない代理認証の構成とする場合、ループ防止機能を無効にする必要があります。その場合は、< Location >に囲まれた範囲内に下記のとおり記述します。

```
AuthProxyLoopCheckEnable off
```

4.3.4 パスワードと Authorization ヘッダー値の復号化機能

パスワードと Authorization ヘッダー値の復号化機能は、OpenAM ポスト認証プラグインと組み合わせて使用します。mod_authproxy ではポスト認証プラグインのプロパティファイルで指定した暗号鍵文字列を、復号鍵文字列として指定します。

復号鍵文字列の定義は Apache のサーバー単位で行うため、< Location >に囲まれた範囲には記述しません。

```
AuthProxyOpenAMDecryptPassword “ポスト認証プラグインで指定した文字列”
```

この定義ではパスワードの復号化を行えます。Authorization ヘッダー値の復号化を行う場合は、Authorization ヘッダー値が入った HTTP ヘッダー名を指定します。OpenAM Agent で指定した名称と同じ値を指定します。

```
AuthProxyAuthorizationName Authorization
```

4.3.5 HTML 登録

mod_authproxy が生成する「自動認証情報送信 HTML」は、デフォルトでは動的に HTML 画面を作成して応答します。しかし、あらかじめ作成・設定しておいた HTML を応答させることも可能です。

mod_authproxy は HTML ファイルを読み込み、下記の文字列の置換処理を行い「自動認証情報送信 HTML」として応答します。

置換前文字列	置換後文字列
%authproxy_action%	アプリケーションの POST 送信先 URL（form タグ action 値または AuthProxySendUrl の設定値）
%AuthProxyLoginData の第一引数 %	AuthProxyLoginData の第一引数の value 値

自動認証情報送信 HTML の例：

```
<html>
<head>
  <title>Login Page</title>
  <script language="JavaScript">
    <!--
    function LoadData()
    {
        document.form.submit();
    }
    //-->
  </script>
</head>
<body onLoad="LoadData();">
```

```
  <form name="form" action="%authproxy_action%" method="POST">
    <INPUT type="hidden" name="LoginUserID" value="%username%">
    <INPUT type="hidden" name="LoginPassword" value="%password%">
  </form>

</body></html>
```

用意した HTML ファイルをサーバー上の任意の場所に配置した上で、AuthProxyHTML-File ディレクティブを用いて、配置したファイルを指定します。

```
AuthProxyHTMLFile /opt/osstech/var/authproxy/authproxy.html
```

4.3.6 ダイレクトに認証情報を送信する代理認証

デフォルトでは、自動認証情報送信 HTML によるモードで代理認証は動作します。ダイレクトに認証情報を送信するモードにしたい場合は、AuthProxyDirectModeEnable on を設定に追加することで、動作モードの切り替えが可能です。

4.4 アプリケーションのログアウトについて

mod_authproxy 導入後のアプリケーションのログアウトについての考慮点を説明します。

通常アプリケーションにはログアウトボタンが用意され、ユーザーがログアウトボタンを押すことでアプリケーションのログアウトが行われます。mod_authproxy 導入した際は、ログアウトボタン押下後の画面遷移を考慮する必要があります。

mod_authproxy の動作仕様により、アプリケーションのログイン画面が応答されると自動的にログインが行われます。そのため「アプリケーションのログアウトボタン操作後にログイン画面が返る画面遷移」では、mod_authproxy による代理認証機能が働き、再度アプリケーションにログインされてしまいます。（図 11）

図 11 のように、ユーザーから見ると「ログアウトボタンを押したのに、再度ログイン成功後の画面が表示される」というように、ログアウトされていないように見える画面遷移となってしまいます。このような画面遷移とならないための対応策として、次の 2 通りの対応が考えられます。

1. アプリケーション側で、ログアウト成功後の画面で「ログイン画面」を返さないように対応する
2. OpenAM Agent の「エージェントログアウト URL」を定義する

1 の方法では、アプリケーションによっては、ログアウト後に遷移する画面の URL を設

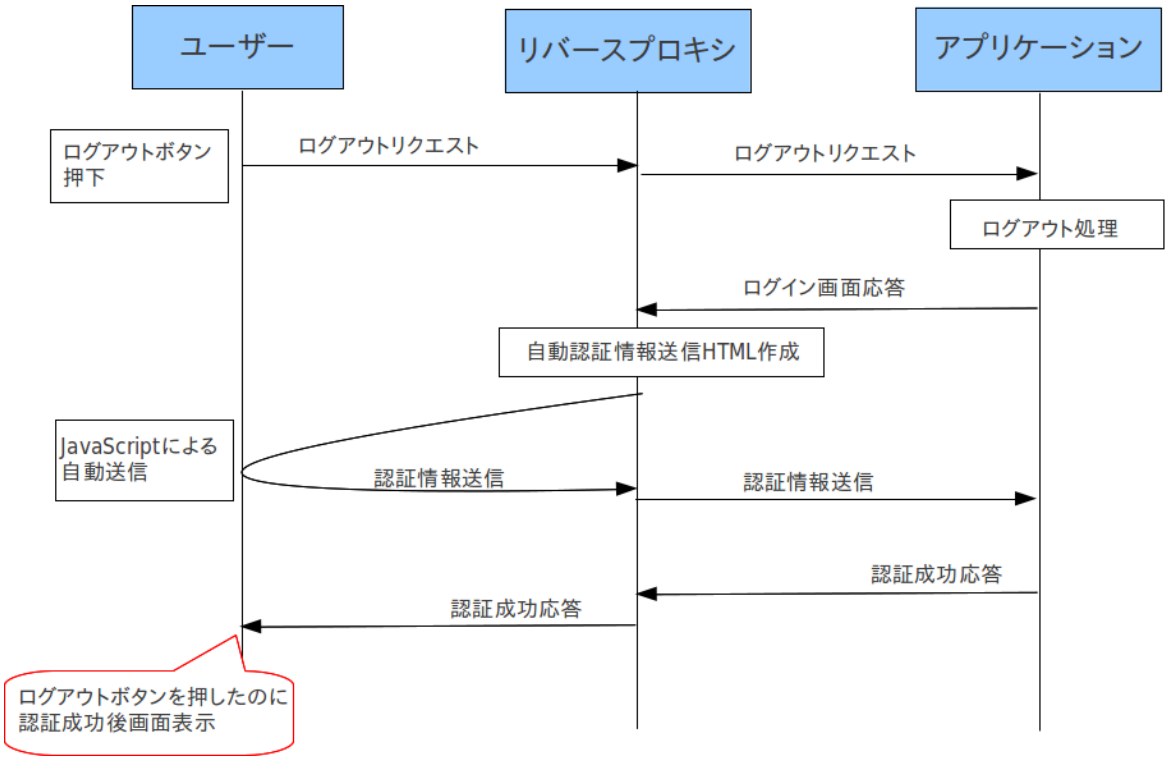

図 11　アプリケーションのログアウトについて

定で定義可能な場合があります。遷移する URL を OpenAM のログアウト画面等、アプリケーションのログイン画面が返らない URL に変更します。

2 の方法では、OpenAM のエージェントの設定で、アプリケーションのログアウト実行時の URL を「エージェントのログアウト URL」として登録します。

この設定を行うことで、アプリケーションのログアウト実行時に OpenAM のログアウトが行われます。注意すべき点として、この方式ではアプリケーションにはログアウトのリクエストが送信されません。OpenAM からはログアウトされますが、アプリケーション上ではログインされたままとなります。そのため必ず、アプリケーションのセッションのタイムアウトを設け、ユーザーのログアウト操作無しでもタイムアウトでログアウトされる構成とします。また、OpenAM のログアウト成功画面でユーザーにブラウザを閉じる旨のメッセージを表示させ、ブラウザの Cookie を削除するよう仕向けることをお勧めします。

mod_authproxy と OpenAM によりシングルサインオンは行えますが、シングルログアウト (1 度の操作ですべてのアプリケーションのログアウト) は行えません。アプリケーション毎にログアウト操作をしてもらう必要があります。しかし、シングルサインオンをしているため、ユーザーはログアウト操作を一度しかしないで良いという認識を持ちます。そのため

ログアウトボタン実行時は OpenAM のログアウトを行うようにし、各アプリケーションではセッションのタイムアウトでユーザーのログアウトとすることを推奨します。

5 リバースプロキシを用いない代理認証

代理認証モジュールは、リバースプロキシの構成にしなくても実現可能です。本章ではリバースプロキシ構成を取らない代理認証について説明します。

5.1 シングルサインオン実現構成

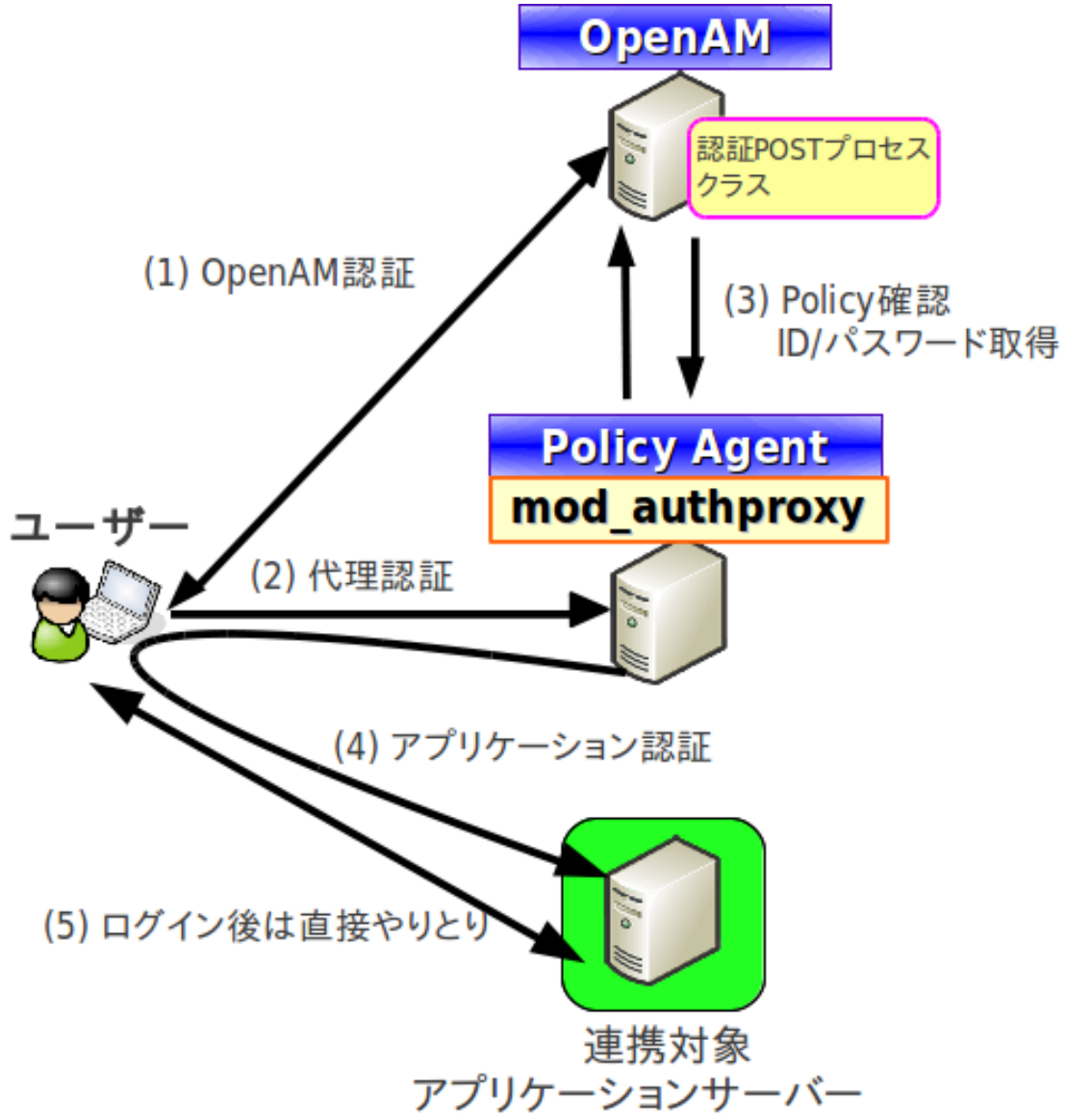

図 12　リバースプロキシ構成を用いない代理認証の構成図

図 12 のように、ユーザーは OpenAM 認証後に代理認証モジュールを導入したサーバーにアクセスし、ユーザーが直接アプリケーションへ認証情報を送ることでシングルサインオンを実現します。アプリケーション認証後は、ユーザーが直接アプリケーションサーバーと通信することになります。

アプリケーションへの最初のアクセスのリンク先を代理認証モジュールを導入したサーバーにすることで実現します。具体的には、ポータルのリンク先、ユーザーへアナウンスするアプリケーションへのリンク先 URL を代理認証モジュールを導入したサーバーの URL にします。

5.2 方式メリット/デメリット

「リバースプロキシ構成」と「リバースプロキシ構成を取らない代理認証」を比較すると次のメリット/デメリットがあります。

	メリット	デメリット
リバースプロキシ構成	OpenAM で認可のコントロールができる アクセスの集中管理が行えるため、誰がどこにアクセスしたかの情報の一元管理を行える	リバースプロキシサーバーに負荷が集中する
リバースプロキシに用いない構成	OpenAM 障害時も直接アプリケーションサーバーにアクセスし、ログインして使うことができる （シングルサインオンせずにログインできる）	シングルサインオンを行えないケースがある （5.4 シングルサインオン不可条件を参照）

なお、リバースプロキシを用いない構成では、アプリケーションへの認証リクエストが mod_authproxy を経由しないため「認証データの暗号化/復号化機能」「リプレイアタック防止機能」「ループ防止機能」の機能は使用出来ません。

5.3 構築方法

リバースプロキシ構成を取らない代理認証では、特定の URL が実行されたら認証情報を仕向けるようにします。

以下の例の構成で説明します。

- OpenAM サーバー：openam.example.co.jp
- 代理認証サーバー：authproxy.example.co.jp
- アプリケーションサーバー: ap1.example.co.jp
- ログイン用 URL:http://ap1.example.co.jp/login.do

5.3.1 ログイン用 URL の決定

アプリケーションサーバーへ認証を行うための自動認証情報送信 HTML 返答用 URL を決定します。ここでは「ap1_login.html」とします。

5.3.2 ap1_login.html コンテンツの配置

ap1_login.html コンテンツをドキュメントルート（デフォルト:/var/www/html）に配置します。コンテンツの中身は下記の通りです。

```
<html><body>認証</body></html>
```

5.3.3 代理認証の設定

「4.1.2 アプリケーションに送信する認証情報」に従い、アプリケーションに送信する情報、URL を調べます。アプリケーションへの認証リクエストがリバースプロキシに来ないため、ループ防止用 Cookie を使用しない設定を行います。例えば、ID/パスワードだけ送信する設定は下記の通りとなります。

```
<Location /ap1_login.html>
    <IfModule mod_filter.c>
        FilterProvider authproxy  authproxy  resp=content-type $text/html
        FilterProtocol authproxy  "change=yes"
        FilterChain authproxy
    </IfModule>
    AuthProxyLoginKeyword "<html>"
    AuthProxyLoginData username %{userid}i
    AuthProxyLoginData password %{authpass}i
    AuthProxyLoopCheckEnable Off
    AuthProxySendUrl http://ap1.example.co.jp/login.do
</Location>
```

上記の mod_filter の設定は Apache2.2 の方法です。Apache2.4 は設定方法が異なります。詳細は「4.2.2 output フィルターの登録」を参照してください。

5.3.4 認証情報にログイン画面のデータが必要な場合

アプリケーションによって、ログイン時に必要な情報としてログイン画面に含まれるデータが必要な場合があります。その場合は mod_proxy を使用し、ログイン画面取得のみリバースプロキシ構成にします。(図 13)

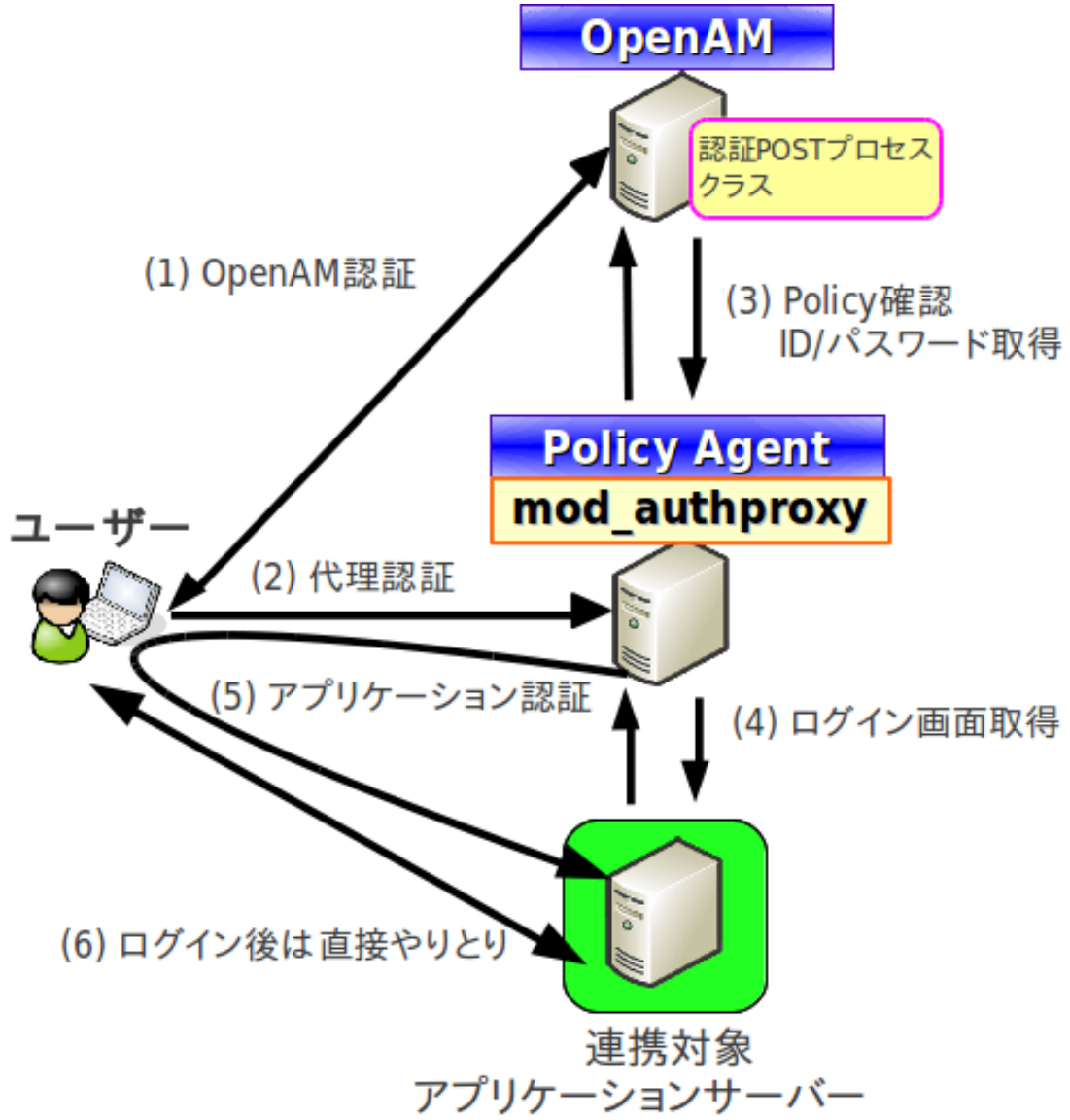

図 13　認証情報にログイン画面のデータが必要な場合

構築方法は、「4.1 アプリケーションのログイン情報の収集」で調査し、「4.2 mod_authproxy の設定」の手順に従って設定項目を決定します。

基本的に同じ手順ですが、「AuthProxySendUrl」ディレクティブを使い、認証情報送信を

アプリケーションサーバーに行うようにします。 Query String が含まれる等ログイン画面内の Action 属性の URL を使用したい場合は「AuthProxySendUrlBase」ディレクティブを用います。

またアプリケーションへの認証リクエストがリバースプロキシに来ないため、ループ防止用 Cookie を使用しない設定「AuthProxyLoopCheckEnable Off」を行います。

5.4 シングルサインオン不可条件

「リバースプロキシを用いない代理認証」には制限があります。アプリケーションが下記の条件に該当するとシングルサインオンを行うことはできません。

1. ログイン画面表示時に Cookie をセットし、認証情報送信時にその Cookie を必要とするアプリケーション
2. ログイン画面表示のソース IP アドレスと、認証情報送信時のソース IP が異なると認証できないアプリケーション

なお、これらの制限はリバースプロキシ構成にはありません。これらの条件に合致する場合はリバースプロキシ構成にしてください。

6 トラブルシューティング

本章では、代理認証オプションの導入を行う際のトラブルシューティングについて記載します。

6.1 Apache のログレベル

mod_authproxy 導入時は、Apache のログレベルを「info」としてください。info レベルにすることで、設定が正しく行われているか等、導入時に必要な情報が出力されます。

mod_authproxy 運用時は、Apache のログレベルを「warn」以上としてください。info レベルでは通常アクセスで多くの情報が出力されるため、システム運用時には不向きです。ユーザーに影響のあるエラーは warn 以上で出力されるため、運用時は「warn」以上とします。

6.2 トラブルシューティング

mod_authproxy を導入してもアプリケーションにログインできない、という場合の調査方法を示します。以下の点が前提となります。

1. リバースプロキシサーバー、OpenAM サーバー、アプリケーションサーバーは構築済みであること。
2. mod_authproxy を導入していない構成で、「OpenAM のログイン」「アプリケーションのログイン」を行うとアプリケーションを利用できること。

上記 2 点が満たされている状態で、mod_authproxy の設定を実施し、Apache のログレベルを info にします。そして、リバースプロキシサーバー経由でアプリケーションのログイン画面にアクセスし、Apache のエラーログを確認します。

6.2.1 確認のポイント

Apache エラーログに以下が出力されていることを確認します。

```
[日時] [info] authproxyOutFilter()
 [プロセス ID] Not match NOTLOGINKEY:[ログイン画面に含まれない文字列設定].
 Request_uri:[URL]
```

または

```
[日時] [info] authproxyOutFilter()
 [プロセス ID] Not match NOTLOGINKEY:[ログイン画面に含まれない文字列設定].
 Request_uri:[URL]
```

上記のログのどちらかが出て入れば、mod_authproxy はアプリケーションのログイン画面を正しく認識しています。上記ログが出ていない場合、そもそも mod_authproxy の output フィルターが動作していません。mod_filter の定義や＜ Location ＞で囲んでいる URL が正しく設定されているか確認します。

下記のどちらかのログが出ている場合は、ログイン画面に含まれる文字列の問題で、mod_authproxy はコンテンツをログイン画面と認識できていません。

```
[日時] [info] authproxyOutFilter()
 [プロセス ID] Not match LOGINKEY:[ログイン画面に含まれる文字列設定].
  Filter END. Request_uri:[URL]
```

または

```
[日時] [info] authproxyOutFilter()
 [プロセス ID] match NOTLOGINKEY:[ログイン画面に含まれる文字列設定].
  Filter END. Request_uri:[URL]
```

mod_authproxy がログイン画面を認識できているがアプリケーションのログインに成功しない場合は、認証情報（POST データ）が足りない可能性が高いです。ブラウザの JavaScript を off にし、自動認証情報送信 HTML の中身を確認し、意図したデータを送信するコンテンツになっているか確認します。

意図したデータを送信するコンテンツであれば、以下のような観点で原因を究明します。

- アプリケーションでログインに関してエラーが出ていないか、
- mod_authproxy のエラーが出ていないか、
- tcpdump 等を取得し、正常にログインできるケースと mod_authproxy 経由でうまくいかないケースのデータを比較。

6.3 エラーメッセージ

mod_authproxy が出力する、warn 以上のエラーログの出力内容を示します。

レベル	出力メッセージ	説明	影響
ERROR	authProxy Key size is “鍵サイズ” bits - should be 256 bits	POST データ暗号化/復号化用鍵を生成した際に鍵サイズが 256bit ではない。	Apache が起動しない。
ERROR	authProxy Key size is “鍵サイズ” bits - should be 128 bits	OpenAM ポスト認証プラグイン復号化用鍵を生成した際に鍵サイズが 128bit ではない。	Apache が起動しない。
ERROR	authProxy input filter authproxy_parse_form_data() Error.res:[“戻り値”] User:[“ユーザー名”]	authproxy_parse_form_data() 関数でのエラー。	エラー画面応答。(画面は使用ハンドラによる)
ERROR	authProxy input filter decrypt Error. User:[“ユーザー名”]	POST データの復号化に失敗した。	エラー画面応答。(画面は使用ハンドラによる)
ERROR	authProxy input filter Login Data Send Timeout:[“経過時間 (秒)”] User:[“ユーザー名”]	リプレイアタック防止機能により、POST データの有効期間が過ぎていた。	エラー画面応答。(画面は使用ハンドラによる)
ERROR	System Error. postsendinputfilter Not found Context. User:[ユーザー名]	Apache 内部エラー	エラー画面応答。(画面は使用ハンドラによる)
ERROR	authproxyOutFilter() ap_run_sub_req() Error. rr_status:[内部処理戻り値] rr->status:[内部処理戻り値] User:[ユーザー名]	ダイレクトに認証情報を送信でエラーが発生	エラー画面応答。(画面は使用ハンドラによる)

レベル	出力メッセージ	説明	影響
WARNING	authproxyfixup() OpenAM Data Decrypt Fail. headername:[“ヘッダー名”] User:[“ユーザー名”]	ポスト認証プラグインで暗号化されたデータの復号化に失敗した。	データが暗号化されたままの値で代理認証が行われる。
WARNING	authproxyfixup() Authoraization Header Decrypt Fail. name:[“ヘッダー名”] User:[“ユーザー名”]	ポスト認証プラグインで暗号化された Authorization ヘッダー値の復号化に失敗した。	Authorization ヘッダー値が暗号化されたままとなる
INFO	authproxyfixup() Not Authorization Header. name:[“ヘッダー名”] User:[“ユーザー名”]	ポスト認証プラグインで暗号化された Authorization ヘッダーが無い。	なし
WARNING	authproxyOutFilter() Not Encrypt. max encrypt value.(name:[“名前”] length:[“サイズ”] User:[“ユーザー名”]	POST データの暗号化で、暗号化元データのサイズが上限値 (4096) を越えたサイズであったため、暗号化を行わなかった。	該当のデータの POST データ暗号化が行われない。
WARNING	authproxy duplicate check Error. User:[“ユーザー名”]	ループ防止機能によりログイン画面と判定したが、自動認証情報送信 HTML の応答をしなかった。	アプリケーションから応答のあった画面を応答

レベル	出力メッセージ	説明	影響
WARNING	authproxy Form action NULL. User:[“ユーザー名”]	Form タグから aciton 属性の値を見つけられなかったため自動認証情報送信 HTML の応答をしなかった。	アプリケーションから応答のあった画面を応答
WARNING	authproxyOutFilter() LIMIT AuthProxyMaxLoginPage-Size. content_size:[“サイズ”] User:[“ユーザー名”]	ログイン画面のサイズが上限値 (AuthProxyMaxLoginPa-geSize) を越えた。	アプリケーションから応答のあった画面を応答
WARNING	mod_authproxy.c setcookie_parse Cookie not ’=’	ログイン画面への応答レスポンスの Set-Cookie でヘッダー値に=が無い。	認証情報 POST 時にこの Set-Cookie が送られない。
WARNING	mod_authproxy.c setcookie_parse Cookie not value	ログイン画面への応答レスポンスの Set-Cookie でヘッダー値に Cookie 値が無い。	認証情報 POST 時にこの Set-Cookie が送られない。
WARNING	authproxyOutFilter() ap_sub_req_lookup_uri() Error. status:[内部ステータスコード] User:[ユーザー名]	Apache 内部エラー	アプリケーションから応答のあった画面を応答

7 mod_authproxy の設定

7.1 ディレクティブ一覧

mod_authproxy で設定可能なディレクティブの一覧は下記のとおりです。

項目名	説明	コンテキスト
AuthProxyLoginKeyword	ログイン画面に含まれる文字列を指定	ディレクトリ
AuthProxyLoginNotKeyword	ログイン画面に含まれる文字列を指定	ディレクトリ
AuthProxyLoginData	アプリケーションに送信する POST データを指定	ディレクトリ
AuthProxyCookieName	authProxy が発行する Cookie の Cookie 名	サーバー
AuthProxyCookieAttribute	authProxy が発行する Cookie の属性を設定	サーバー
AuthProxyMaxLoginPageSize	ログイン画面の最大サイズを指定	ディレクトリ
AuthProxyEncryptPassword	POST データを暗号化するための鍵文字列	サーバー
AuthProxyFormName	ログイン画面内の Form タグの name 値	ディレクトリ
AuthProxySendUrl	POST データを送信する URL	ディレクトリ
AuthProxySendUrlBase	POST データを送信する URL のベースになる値	ディレクトリ
AuthProxyHTMLFile	自動認証情報送信 HTML ファイルを指定	ディレクトリ
AuthProxyLoginDataTTL	認証データの有効期間	サーバー
AuthProxyAddType	自動認証情報送信 HTML 応答時のレスポンスヘッダーの Content-Type を指定	ディレクトリ

項目名	説明	コンテキスト
AuthProxyOpenAMDecryptPassword	ポスト認証プラグインで暗号化されたデータを復号化するための鍵文字列を指定	サーバー
AuthProxyAuthorizationName	ポスト認証プラグインで暗号化された Authorization ヘッダー値が入った HTTP ヘッダー名を指定	サーバー
AuthProxyLoopCheckEnable	ループ防止機能の有効有無を設定します。有効にするとループ防止機能判定用の Cookie が発行されます。	ディレクトリ
AuthProxyDirectModeEnable	直接アプリケーションサーバーに認証情報を送信するモードで動作します。	ディレクトリ
AuthProxyPOSTCheckName	リクエストの POST データを置換するパラメーター名を指定	ディレクトリ
AuthProxyLibxmlErrorEnable	libxml2 で出力されるメッセージの出力有無を指定	ディレクトリ

本モジュールのコンテキスト「サーバー」のディレクティブは、「AuthProxyLoginDataTTL」と「AuthProxyAuthorizationName」を除いてサーバ設定ファイルで一つのみ設定可能となります。＜ VirtualHost ＞に囲まれた中に記載しても無視します。

7.2 ディレクティブ詳細

各ディレクトリの詳細な説明を記します。

7.2.1 AuthProxyLoginKeyword ディレクティブ

説明	ログイン画面に含まれる文字列を指定
構文	AuthProxyLoginKeyword “文字列”
デフォルト	なし
コンテキスト	ディレクトリ
設定例	`AuthProxyLoginKeyword “<Title >Login</Title>”`

【詳細説明】 ログイン画面と判定するためにログイン画面に含まれる文字列を指定します。複数行に渡って記載した場合は、and 条件となります。（設定した全ての文字列が含まれていたらログイン画面と判定します。）or 条件の指定はできません。

ログイン画面の文字コードと同じ文字コードで設定ファイルに記載することで、日本語を含む文字列も設定可能です。UTF-8、Shift-JIS、EUC-JP で動作確認をしています。

7.2.2 AuthProxyLoginNotKeyword ディレクティブ

説明	ログイン画面に含まれない文字列を指定
構文	AuthProxyLoginNotKeyword “文字列”
デフォルト	なし
コンテキスト	ディレクトリ
設定例	`AuthProxyLoginNotKeyword “<Title >contents</Title>”`

【詳細説明】 ログイン画面を判定するための、ログイン画面に含まれない文字列を指定します。複数行に渡って記載した場合は、and 条件となります。（設定した全ての文字列が含まれていたらログイン画面でないと判定します）or 条件の指定はできません。ログイン画面の文字コードと同じ文字コードで設定ファイルに記載することで、日本語を含む文字列も設定可能です。UTF-8、Shift-JIS、EUC-JP で動作確認をしています。AuthProxyLoginKeyword と両方指定することが可能です。その場合すべて and 条件となります。

7.2.3 AuthProxyLoginData ディレクティブ

説明	アプリケーションに送信する POST データを指定
構文	`AuthProxyLoginData “name` **値** `” “value` **値** `” [encrypt|noencrypt] [`**デフォルト値**
デフォルト	なし
コンテキスト	ディレクトリ
設定例	`AuthProxyLoginData uname %{userid}i encrypt`

【詳細説明】 認証情報として送信する POST データ (name=value 形式) の設定を行います。name 値は、そのまま POST データの name になります。value 値は以下の 4 パターンの設定が可能です。

1. 固定のデータ：value 値に固定データをそのまま記述

2. HTTP ヘッダーの値：value 値に %{HTTP ヘッダー名}i と指定
3. Form タグ input タグの value 値：value 値に %{name 名}n と指定
4. Apache 環境変数の値：value 値に %{環境変数名}e と指定

「Apache 環境変数の値」は osstech-mod_authproxy-1.0-21 以降のバージョンで指定可能です。

第三引数, 第四引数は任意です。

第三引数は暗号化有無を指定します。該当の value 値を暗号化する場合に encrypt と記述します。省略した場合や noencrypt と指定すると暗号化されません。暗号化処理を行う場合は、復号化 input フィルターの登録が必ず必要です。” ダイレクトに認証情報を送信する代理認証” モードで動作している場合は、encrypt を指定しても暗号化して送られません。暗号化できる値には 4096byte の上限値があります。上限値を変更することはできません。

第四引数は、value 値が空の場合のデフォルト値を指定します。value 値が空の場合はデフォルト値が POST データで送信されます。デフォルト値は value 値のパターンで HTTP ヘッダーの値を設定した場合のみ有効です。

【設定例】

```
AuthProxyLoginData test1 “testvalue”
AuthProxyLoginData test2 “%{User-Agent}i”
AuthProxyLoginData test3 “%{uid}n”
AuthProxyLoginData test4 “%{Accept-Language}i noencrypt en”
```

【ログイン画面】

```
<html>
  <body>
    <form name='test' method='post' action='/login.pl'>
      <input type=hidden name='uid' value='user01'>
    </form>
  </body>
</html>
```

【アクセスユーザーの HTTP ヘッダー】

```
GET /login.html HTTP/1.0
User-Agent: Firefox
```

上記のような定義の場合、代理認証で送信される POST データは以下のようになります。

```
test1=testvalue&test2=Firefox&test3=user01&test4=en
```

7.2.4 AuthProxyCookieName ディレクティブ

説明	authProxy が発行する Cookie の Cookie 名を指定
構文	AuthProxyCookieName “**文字列**”
デフォルト	authproxy
コンテキスト	サーバー
設定例	AuthProxyCookieName “authproxy”

【詳細説明】 ループ処理の防止のため、自動応答コンテンツ応答時に Set-Cookie を行います。その際の Cookie 名を指定します。

7.2.5 AuthProxyCookieAttribute ディレクティブ

説明	authProxy が発行する Cookie の属性を指定
構文	AuthProxyCookieAttribute “**文字列**”
デフォルト	authproxy
コンテキスト	サーバー
設定例	AuthProxyCookieAttribute “path=/;secure;httponly”

【詳細説明】 ループ処理の防止のために発行する Cookie の属性を設定します。本ディレクティブが省略された場合の Set-Cookie の属性は「path=/」となります。

7.2.6 AuthProxyMaxLoginPageSize ディレクティブ

説明	ログイン画面の最大サイズを指定
構文	AuthProxyMaxLoginPageSize **数字**
デフォルト	1064960
コンテキスト	ディレクトリ
設定例	AuthProxyMaxLoginPageSize 1064960

【詳細説明】 ログイン画面の最大サイズを指定します。ログイン画面判定処理時にこのサイズを越えるコンテンツである場合、ログインキーワードで一致してもログイン画面と判定しません。内部処理で大きなコンテンツのメモリ確保を行わないようにするための定義であり、通常は設定を行う必要はありません。ログイン画面が 10Mbyte(1064960) 以上である場合は設定が必要です。

7.2.7 AuthProxyEncryptPassword ディレクティブ

説明	POST データを暗号化するための鍵文字列
構文	`AuthProxyEncryptPassword “文字列”`
デフォルト	1234567890
コンテキスト	サーバー
設定例	`AuthProxyEncryptPassword “xxxxxxxxxxxxx”`

【詳細説明】 アプリケーションへ送信する POST データに暗号化を行う指定をした場合、暗号化/復号化に使用する鍵文字列を設定します。

7.2.8 AuthProxyFormName ディレクティブ

説明	ログイン画面内の Form タグの name 値
構文	`AuthProxyFormName “文字列”`
デフォルト	なし
コンテキスト	ディレクトリ
設定例	`AuthProxyFormName testauth`

【詳細説明】 ログイン画面内の form タグが複数ある場合に、ログイン用の FORM タグと認識するための name 値を指定します。名前と一致した画面内の form タグ input タグの値や aciton 属性を使用します。

ログイン画面に form タグが複数ある場合に指定します。一つしかない場合は、本ディレクトリの設定を省略することができます。

7.2.9 AuthProxySendUrl ディレクティブ

説明	POST データを送信する URL

構文	AuthProxySendUrl URL
デフォルト	なし
コンテキスト	ディレクトリ
設定例	AuthProxySendUrl /login.php

【詳細説明】 POST データを送信する URL を指定できます。省略した場合は、Form タグ内の action 属性の値を使用します。Form タグが複数あり、name 値がない場合に送信 URL を指定します。

7.2.10 AuthProxySendUrlBase ディレクティブ

説明	POST データを送信する URL のベースになる値
構文	AuthProxySendUrlBase URL
デフォルト	なし
コンテキスト	ディレクトリ
設定例	AuthProxySendUrlBase http://ap.example.co.jp

【詳細説明】 POST データの送信先の URL を Form タグ内の action 属性から取得した場合、送信先 URL が「本ディレクティブの値」＋「action 属性の値」となります。本設定はリバースプロキシを用いない代理認証の構成とした際、POST データの送信先 URL をアプリケーションのログイン画面内から取得したい場合に使用します。

本ディレクティブにアプリケーションサーバーの http(s)://～から始まる絶対パスを記載することで、リバースプロキシを用いない代理認証の構成でも POST データを送信する URL を正しく設定することができます。

7.2.11 AuthProxyHTMLFile ディレクティブ

説明	自動認証情報送信 HTML ファイルを指定
構文	AuthProxyHTMLFile **ファイル名**
デフォルト	なし
コンテキスト	ディレクトリ
設定例	AuthProxyHTMLFile /tmp/test.txt

【詳細説明】 自動認証情報送信 HTML ファイルを指定します。省略した場合は、authProxy が動的に作成した HTML となります。HTML ファイルでは、action や POST データ部分の記述を置換して応答します。(「4.3.5 HTML 登録」 参照)

7.2.12 AuthProxyLoginDataTTL ディレクティブ

説明	認証データの有効期間
構文	`AuthProxyLoginDataTTL` **数値**
デフォルト	60
コンテキスト	ディレクトリ
設定例	`AuthProxyLoginDataTTL 60`

【詳細説明】 認証データの有効期間を秒単位で設定します。自動認証情報送信 HTML 生成後、本設定以内の時間に POST されなかった場合、エラーが返ります。返答されるエラー画面は動作するハンドラによって変わり、リバースプロキシ構成で mod_proxy_html の場合は 400 エラーが返ります。本設定は POST データを一つでも暗号化する設定にした場合に有効です。なお、0 を設定すると無制限となり、有効期間のチェックをしません。

7.2.13 AuthProxyAddType ディレクティブ

説明	自動応答時のレスポンスヘッダーの Content-Type
構文	`AuthProxyAddType` **“文字列”**
デフォルト	なし
コンテキスト	ディレクトリ
設定例	`AuthProxyAddType “text/html; charset=UTF-8”`

【詳細説明】 自動認証情報送信 HTML 応答時のレスポンスヘッダーの Content-Type を指定します。指定がない場合は本モジュールでセットしないため、Apache ハンドラから受けとった Content-Type がそのままセットされます。AuthProxyHTMLFile にて指定した HTML ファイルの文字コードを設定するためのディレクティブです。

7.2.14 AuthProxyOpenAMDecryptPassword ディレクティブ

説明	ポスト認証プラグインで暗号化されたデータを復号化するための鍵文字列

構文	AuthProxyOpenAMDecryptPassword **“文字列”**
デフォルト	AuthProxy
コンテキスト	サーバー
設定例	AuthProxyOpenAMDecryptPassword xxxxxxxxxx

【詳細説明】 ポスト認証プラグインで暗号化されたデータを復号化するための鍵文字列として、ポスト認証プラグインのプロパティファイルで設定した暗号鍵文字列と同じ値を指定します。弊社開発のポスト認証プラグインを導入していない場合は、本設定は不要です。

7.2.15 AuthProxyAuthorizationName ディレクティブ

説明	ポスト認証プラグインで暗号化された Authorization ヘッダー値が入った HTTP ヘッダー名
構文	AuthProxyAuthorizationName **“文字列”**
デフォルト	なし
コンテキスト	サーバー
設定例	AuthProxyAuthorizationName AUTHORIZATIONDATA

【詳細説明】 OpenAM Agent の「セッション属性処理」→「セッション属性マップ」で追加した” 対応するマップ値” の HTTP ヘッダー名を指定します。「本設定項目の設定値」と「HTTP ヘッダー名」を前方一致で比較し、一致した HTTP ヘッダーに対してデータの復号化処理を行います。アプリケーション単位に BASIC 認証を複数用意する場合は” 対応するマップ値” に対して<VirtualHost>単位で異なる HTTP ヘッダー名を定義します（下記「設定例」参照)。

【設定例】

ポスト認証プラグインのプロパティファイル

- BasicAuth01=AuthorizationData01,uid,Password
- BasicAuth02=AuthorizationData02,mail,Description
- BasicAuth03=AuthorizationData03,uid,"password"

OpenAM Agent の「セッション属性処理」→「セッション属性マップ」

マップキー	マップ値
AuthorizationData	x-AuthorizationData-ap1
AuthorizationData01	x-AuthorizationData-ap2

マップキー	マップ値
AuthorizationData02	x-AuthorizationData-ap3
AuthorizationData03	x-AuthorizationData-ap4

httpd.conf(抜粋)

```
LoadModule ssl_module modules/mod_ssl.so
<VirtualHost *:80>
    ServerName ap1.example.co.jp
    AuthProxyAuthorizationName x-AuthorizationData-ap1
    RequestHeader set Authorization %{HTTP:x-AuthorizationData-ap1}s
</VirtualHost>
<VirtualHost *:80>
    ServerName ap2.example.co.jp
    AuthProxyAuthorizationName x-AuthorizationData-ap2
    RequestHeader set Authorization %{HTTP:x-AuthorizationData-ap2}s
</VirtualHost>
<VirtualHost *:80>
    ServerName ap3.example.co.jp
    AuthProxyAuthorizationName x-AuthorizationData-ap3
    RequestHeader set Authorization %{HTTP:x-AuthorizationData-ap3}s
</VirtualHost>
<VirtualHost *:80>
    ServerName ap4.example.co.jp
    AuthProxyAuthorizationName x-AuthorizationData-ap4
    RequestHeader set Authorization %{HTTP:x-AuthorizationData-ap4}s
</VirtualHost>
```

なお、BASIC 認証のシングルサインオンを行わない場合やポスト認証プラグインを導入していない場合、本設定は不要です。

7.2.16 AuthProxyLoopCheckEnable ディレクティブ

説明	ループ防止機能を使用有無を指定
構文	`AuthProxyLoopCheckEnable [on|off]`
デフォルト	on
コンテキスト	ディレクトリ
設定例	`AuthProxyLoopCheckEnable off`

【詳細説明】 off にするとループ防止機能の Cookie を発行せず、ループ防止機能 Cookie のチェックをしません。

7.2.17 AuthProxyDirectModeEnable ディレクティブ

説明	ダイレクト認証情報送信モードの使用有無を指定
構文	`AuthProxyDirectModeEnable [on|off]`
デフォルト	off
コンテキスト	ディレクトリ
設定例	`AuthProxyDirectModeEnable off`

【詳細説明】 on にするとダイレクトに認証情報を送信するモードで動作します。

7.2.18 AuthProxyPOSTCheckName ディレクティブ

説明	置換する POST データの name を指定
構文	`AuthProxyPOSTCheckName "name` **値**`" [value` **値**`]`
デフォルト	なし
コンテキスト	ディレクトリ
設定例	`AuthProxyPOSTCheckName userid`

【詳細説明】 ユーザーから入力された POST パラメーター値を置換したい場合に置換する POST パラメーター名を指定します。代理認証先のパスワードが固定であるケースで、ユーザーが POST パラメーターのユーザー ID 部分を変更できないようにするために使用します。

本設定を行うとリクエストの POST データに一致するパラメーターがあれば値が置換されます。置換後の文字列は AuthProxyLoginData で設定した値になります。置換後の文字列を取得できなかった場合は第二引数の [value 値] に置換されます。 [value 値] は省略可能です。

本機能は osstech-mod_authproxy-1.0-22 以降で使用可能です。

7.2.19 AuthProxyLibxmlErrorEnable ディレクティブ

説明	libxml2 のメッセージ出力有無を指定
構文	`AuthProxyLibxmlErrorEnable on|off`

デフォルト	off
コンテキスト	ディレクトリ
設定例	AuthProxyLibxmlErrorEnable on

【詳細説明】 on にすると libxml2 で出力されるメッセージを標準エラー出力 (Apache エラーログ) に出力します。代理認証の検証やテストの際に libxml2 エラーの内容を確認する場合に on と設定します。

本機能は osstech-mod_authproxy-1.0-23 以降で使用可能です。

7.3 Apache 環境変数

mod_authproxy には、代理認証を動作させないようにする Apache 環境変数「noauthproxy」が用意されています。この環境変数がセットされていると、mod_authproxy は動作しません。

例えば、SetEnvIf ディレクティブと組み合わせて、「特定の端末 (ソース IP により判定) からのアクセスに対しては代理認証を動作させない」といった動作が可能です。

```
SetEnvIf Remote_Addr ^172\.16\.17\.18$ noauthproxy=1
```

本機能は osstech-mod_authproxy-1.0-19 以降で使用可能です。

7.3.1 PolicyAgent でセットした HTTP ヘッダー値を使用して制御する

SetEnvIf ディレクティブでは、Policy Agent でセットした HTTP ヘッダー値によって Apache 環境変数をセットすることが出来ません。 これは、Policy Agent より先に mod_setenvif モジュールが動作するためです。

PolicyAgent でセットした HTTP ヘッダー値を使用して代理認証の動作有無を制御する場合は、mod_rewrite を使用します。

【設定例】(ユーザー名が osstechtest01 の場合は代理認証は動作させない場合)

```
<Location /login.php>
    RewriteEngine on
    RewriteCond %{HTTP:userid} ^osstechtest01$
    RewriteRule ^. - [env=noauthproxy:1]

    FilterProvider authproxy  authproxy  resp=content-type $text/html
    FilterProtocol authproxy  "change=yes"
    FilterChain authproxy
    AuthProxyLoginKeyword “<!-- 8fb6f57c3e3993407d48e985e423a72c6dba36fe -->”
```

```
    AuthProxyLoginData uname %{userid}i
    AuthProxyLoginData pass %{authpass}i
    AuthProxyLoginData xoops_redirect %{xoops_redirect}n
    AuthProxyLoginData op %{op}n
    AuthProxyFormName loginA
</Location>
```

mod_rewrite は、必ず＜ Location ＞ディレクティブで囲んだ中に定義します。

8 [ご参考] リバースプロキシ設定例

本章では、Apache にリバースプロキシ構成を構築する設定例を示します。ここではアプリケーションの FQDN をリバースプロキシに割り当て、ユーザーはアプリケーションの FQDN でリバースプロキシにアクセスする構成を示します。

8.1 構成図

本章で設定例を示すサーバーの構成図を下記に記します。

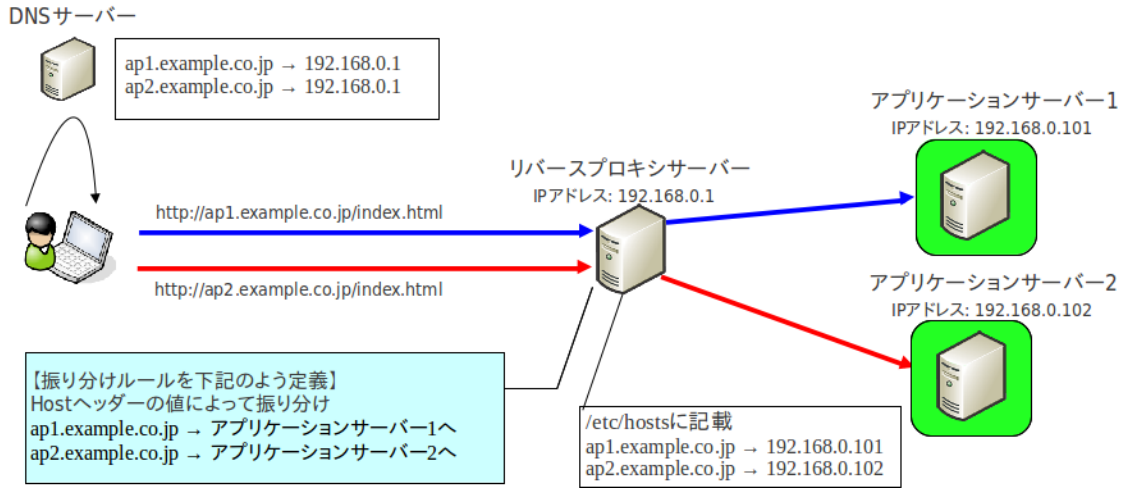

図 14　例で示す設定ファイルの構成

8.2 設定例

「8.1 構成図」の構成で構築した際の、リバースプロキシサーバーの Apache 設定例を示します。リバースプロキシサーバー構築の参考にしてください。

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so

NameVirtualHost *:80
<VirtualHost *:80>
    DocumentRoot "/var/www/html"
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerName app1.example.co.jp
    ProxyPass / http://app1.example.co.jp/
</VirtualHost>

<VirtualHost *:80>
    ServerName app2.example.co.jp
    ProxyPass / http://app2.example.co.jp/
</VirtualHost>
```

9 変更履歴

- 2012 年 6 月 14 日 リビジョン 1.0
  - 初版作成
- 2012 年 8 月 24 日 リビジョン 1.1
  - ディレクティブ名の誤字修正
- 2012 年 9 月 03 日 リビジョン 1.2
  - 「5.1 ディレクティブ一覧」にコンテキスト” サーバー” について加筆
- 2012 年 10 月 15 日 リビジョン 1.3
  - 全体的に認証ポストプロセスクラスの説明と設定方法を追加
- 2012 年 10 月 25 日 リビジョン 1.4
  - 「5. リバースプロキシを用いない代理認証」を追加
  - 「8. [参考] リバースプロキシ設定例」を追加
- 2013 年 1 月 29 日 リビジョン 1.5
  - 「AuthProxySendUrlBase」および「AuthProxyCookieAttribute」ディレクティブを追加
- 2013 年 5 月 20 日 リビジョン 1.6
  - 「3.2 認証 POST プロセスクラスのプロパティファイルの設定」に BasicAuthXX 項目および説明文の追加
  - 「7.2.15 AuthProxyAuthorizationName ディレクティブ」に加筆
- 2013 年 6 月 6 日 リビジョン 1.7
  - 「7.2.15 AuthProxyAuthorizationName ディレクティブ」を修正
- 2014 年 9 月 12 日 リビジョン 1.8
  - AuthProxyDirectModeEnable ディレクトリを追加
  - 「mod_authproxy の動作説明」に” ダイレクトに認証情報を送信する代理認証” の説明を追加
- 2015 年 1 月 7 日 リビジョン 1.9
  - 目次及び改版履歴の修正
- 2015 年 2 月 13 日 リビジョン 2.0
  - RHEL7/CentOS7(Apache2.4) 対応 mod_filter の設定方法を追記
  - スタイルと書式設定を修正
- 2017 年 1 月 13 日 リビジョン 2.1
  - OpenAM13 の設定方法を追記

- 2017 年 6 月 9 日 リビジョン 2.2
  - 「7.3 Apache 環境変数」を追記
- 2018 年 4 月 3 日 リビジョン 2.3
  - 全体的に誤字や表現を修正
- 2018 年 8 月 10 日 リビジョン 2.4
  - AuthProxyLoginData ディレクティブの環境変数対応を追記
- 2018 年 8 月 13 日 リビジョン 2.5
  - AuthProxyPOSTCheckName ディレクティブを追加
- 2018 年 8 月 20 日 リビジョン 2.6
  - AuthProxyLibxmlErrorEnable ディレクティブを追加
- 2019 年 3 月 15 日 リビジョン 2.7
  - 「3.2 認証 POST プロセスクラスのプロパティファイルの設定」の BasicAuthXX 項目でパスワードに固定文字列を設定できる説明文の追加
  - AuthProxyAuthorizationName ディレクティブの設定例を追加