Red Hat JBoss Enterprise Application Platform 7.3

Kerberos による SSO のセットアップ方法

Kerberos を使用して Red Hat JBoss Enterprise Application Platform へのシングルサインオンユーザーアクセスを設定して管理するための手順

Last Updated: 2023-02-28

Red Hat JBoss Enterprise Application Platform 7.3 Kerberos による SSO のセットアップ方法

Kerberos を使用して Red Hat JBoss Enterprise Application Platform へのシングルサインオンユーザーアクセスを設定して管理するための手順

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

本書の目的は、Red Hat JBoss Enterprise Application Platform 内における Kerberos によるシングルサインオン (SSO) を検証し、JBoss EAP で Kerberos を用いた SSO の実用的な設定例を提供することです。本書では Kerberos を用いた SSO について深く掘り下げ、JBoss EAP 内でのセットアップおよび設定方法を説明します。本書をお読みいただく前に、Red Hat JBoss Enterprise Application Platform のセキュリティーアーキテクチャーをお読みいただき、このドキュメントに記載されている SSO および Kerberos の情報を理解できるようにしてください。また、本書では JBoss EAP CLI インターフェイスを使用して設定の変更を行います。スタンドアロン JBoss EAP

インスタンスと JBoss EAP ドメインの両方で CLI を使用する方法の詳細は、JBoss EAP 管理 CLI ガイドを参照してください。本書をお読みいただくと、SSO および Kerberos、JBoss EAP との関係、および設定方法を深く実践的に理解することができます。

目次

第1章 KERBEROS を用いた SSO の検証

1.1. SSO および KERBEROS とは

シングルサインオン (SSO) および Kerberos の基本的な情報は、JBoss EAP セキュリティーアーキテクチャー の**シングルサインオン**を参照してください。

1.2. KERBEROS のコンポーネント

Kerberos は、秘密鍵の暗号化を使用してクライアント/サーバーアプリケーションのユーザー認証を可能にするネットワークプロトコルです。通常、Kerberos はネットワーク上のデスクトップユーザーの認証に使用されますが、追加のツールを使用すると、web アプリケーションのユーザー認証に使用でき、複数の web アプリケーションに SSO を提供することができます。そのため、デスクトップネットワーク上で認証済みのユーザーは、再認証しなくても web アプリケーションのセキュアなリソースにシームレスにアクセスできます。ユーザーはデスクトップベースの認証メカニズムを使用して認証され、認証されたユーザーの認証トークンまたはチケットは web アプリケーションによっても使用されるため、この概念はデスクトップベースの SSO と呼ばれます。この SSO メカニズムは、すべてブラウザーを使用してユーザーの認証とトークンの発行を行うブラウザーベースの SSO など、他の SSO メカニズムとは異なります。

Kerberos プロトコルは、認証および承認で使用する複数のコンポーネントを定義します。

チケット

チケット は、プリンシパルに関する認証および承認決定を発行および実行するために Kerberos が使用するセキュリティートークンの形式です。

認証サービス (AS)

認証サービス (AS) は、プリンシパルが最初にネットワークにログインするときにプリンシパルを確認します。認証サービスは、チケット保証チケット (TGT) の発行を行います。TGT は、チケット保証サービス (TGS) の認証に必要で、セキュアなサービスおよびリソースに再度アクセスするときにも必要になります。

チケット保証サービス (TGS)

チケット保証サービス (TGS) は、サービスチケットと特定のセッション情報を、プリンシパルとアクセスするターゲットサーバーに発行します。これは 、プリンシパルによって提供される TGT と接続先の情報を基にします。このサービスチケットとセッション情報は、接続先への接続を確立し、セキュアなサービスまたはリソースへアクセスするために使用されます。

キー配布センター (KDC)

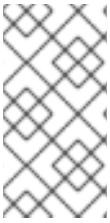
キー配布センター (KDC) は、TGS と AS の両方を格納するコンポーネントです。KDC、クライアントまたはプリンシパル、およびサーバーまたはセキュアなサービスの 3 つのコンポーネントは、Kerberos 認証の実行に必要です。

チケット保証チケット (TGT)

チケット保証チケット (TGT) は、AS がプリンシパルに発行するチケットのタイプです。プリンシパルがユーザー名とパスワードを使用して AS に対して認証されると、TGT が付与されます。TGT はクライアントによってローカルでキャッシュされますが、KDC のみが読み取れ、クライアントは読み取りできないように暗号化されます。これにより、AS は TGS によって使用される TGT の承認データや他の情報をセキュアに保存することができ、TGS はこのデータを使用して承認決定を行うことができます。

サービスチケット (ST)

service ticket (ST) は、TGT と目的の接続先を基にして TGS がプリンシパルに発行するチケットのタイプです。プリンシパルは TGS に TGT を提供し、TGS は TGT の承認データを基にしてプリンシパルが接続先にアクセスしたことを検証します。検証に成功すると、クライアントと、セキュアなサービスまたはリソースが含まれるサーバーである接続先サーバーの両方に対して、TGS は ST をクライアントに発行します。これにより、クライアントは接続先サーバーへのアクセスが許可されます。また、クライアントによってキャッシュされ、クライアントとサーバーの両方が読み取り可能な ST には、クライアントとサーバーがセキュアに通信できるようにするセッション情報も含まれています。



注記

Kerberos とネットワークの DNS 設定には密な関係があります。たとえば、実行しているホストの名前を基にクライアントが KDC にアクセスする場合、仮定を行います。そのため、Kerberos 設定だけでなくすべての DNS 設定を適切に行い、クライアントが接続できるようにすることが重要になります。

1.3. その他のコンポーネント

JBoss EAP で Kerberos SSO を有効にするには、Kerberos コンポーネントの他にも複数の項目が必要になります。

1.3.1. SPNEGO

SPNEGO (Simple and Protected GSS_API Negotiation Mechanism)は Web アプリケーションで使用するために Kerberos ベースのシングルサインオン環境を拡張するメカニズムを提供します。

SPNEGO は、クライアントアプリケーションによって使用される認証方法で、クライアントアプリケーション自体をサーバーに対して認証します。この技術は、相互の通信を試みるクライアントアプリケーションとサーバーがお互いの認証プロトコルを把握していない場合に使用されます。SPNEGO は、クライアントアプリケーションとサーバー間の共通の GSSAPI メカニズムを判断し、その後のセキュリティー操作をすべてディスパッチします。

web ブラウザーなどのクライアントコンピューター上のアプリケーションが web サーバーの保護されたページにアクセスしようとすると、サーバーは承認が必要であることを伝えます。その後、アプリケーションは Kerberos KDC からのサービスチケットを要求します。チケットの取得後、アプリケーションはこのチケットを SPNEGO 向けにフォーマットされたリクエストにラップし、ブラウザーを介して web アプリケーションに返信します。デプロイされた web アプリケーションを実行している Web コンテナーはリクエストをアンパックし、チケットを認証します。認証に成功するとアクセスが許可されます。

SPNEGO は、Red Hat Enterprise Linux に含まれる Kerberos サービスや Microsoft Active Directory には不可欠な Kerberos サーバーなど、全タイプの Kerberos プロバイダーと動作します。

1.3.2. JBoss Negotiation

JBoss Negotiation は、JBoss EAP に同梱されるフレームワークで、オーセンティケーターと JAAS ログインモジュールを提供し、JBoss EAP で SPNEGO をサポートします。JBoss Negotiation は、レガシー **security** サブシステムとレガシーコア管理認証のみと使用されます。JAAS ログインモジュールの詳細は、JBoss EAP セキュリティーアーキテクチャー の 宣言型セキュリティーと JAAS および**セキュリティードメイン**を参照してください。

注記

JBoss Negotiation を使用して、REST web サービスなどの一部のアプリケーションをセキュアにする場合、1つまたは複数のセッションが作成され、クライアントがリクエストを行うと、それらのセッションがタイムアウト期間 (デフォルトでは 30 分) 開かれることがあります。Basic 認証を使用してアプリケーションをセキュアにした場合はセッションを開いたままにしないことが想定されるため、Basic 認証で想定される動作とは異なります。JBoss Negotiation はセッションを使用してネゴシエーション/接続の状態を維持するよう実装されるため、このようなセッションの作成は想定される動作です。

1.4. KERBEROS の統合

Kerberos は、Red Hat Enterprise Linux などの Linux ディストリビューションを含む多くのオペレーティングシステムと統合されています。また、Kerberos は Microsoft Active Directory には不可欠で、Red Hat Directory Server および Red Hat IDM によってサポートされます。

1.5. JBOSS EAP での KERBEROS による SSO の提供

Kerberos は、クライアントおよびサーバーによって使用されるチケットを KDC から発行してデスクトップベースの SSO を提供します。JBoss EAP はこれと同じチケットを独自の認証および承認プロセスで使用して、既存のプロセスと統合することができます。JBoss EAP がどのようにこのチケットを再使用するかを理解する前に、チケットがどのように発行され、JBoss EAP がないデスクトップベースの SSO で認証および承認がどのように Kerberos と動作するかを理解することが重要です。

1.5.1. デスクトップベース SSO での Kerberos による認証および承認

Kerberos は、認証および承認を提供するため、サードパーティーの KDC に依存してクライアントにアクセスするサーバーの認証および承認決定を提供します。決定は 3 つの手順で行われます。

1. 認証の交換
   プリンシパルが最初にネットワークにアクセスする場合またはチケット保証チケット (TGT) なしでセキュアなサービスにアクセスする場合、プリンシパルのクレデンシャルを使用して認証サービス (AS) に対して認証することが要求されます。AS はユーザー提供のクレデンシャルを設定済みのアイデンティティーストアと照合します。認証に成功したら、クライアントによってキャッシュされる TGT がプリンシパルに発行されます。TGT には一部のセッション情報も含まれているため、クライアントと KDC の今後の通信も保証されます。

2. チケットの付与または承認、交換
   プリンシパルに TGT が発行されたら、プリンシパルはセキュアなサービスまたはリソースにアクセスします。プリンシパルはチケット保証サービス (TGS) にリクエストを送信し、KDC によって発行された TGT を渡し、特定の接続先に対するサービスチケット (ST) を要求します。TGS はプリンシパルが提供した TGT をチェックし、要求したリソースにアクセスできる適切な権限があることを検証します。検証に成功したら、TGS はプリンシパルがその特定の接続先にアクセスするための ST を発行します。また、TGS はクライアントと接続先サーバーの両方に対してセッション情報を作成し、クライアントと接続先サーバー間のセキュアな接続を可能にします。このセッション情報は、クライアントとサーバーが、以前のトランザクションで KDC によって個別に提供された長期キーを使用して、独自のセッション情報のみを復号化できるよう個別に暗号化されます。その後、TGS はクライアントとサーバー両方のセッション情報が含まれる ST でクライアントに応答します。

3. サーバーへのアクセス
   これで、プリンシパルはセキュアなサービスの ST と、サーバーとセキュアに通信できるメカニズムを取得したため、クライアントは接続を確立し、セキュアなリソースへアクセスできます。クライアントは最初に ST を接続先サーバーに渡します。この ST には、その接続先用に TGS から受け取ったセッション情報のサーバーコンポーネントが含まれています。サーバー

は、KDC からの長期鍵を使用して、クライアントが渡したセッション情報を復号化します。復号化できたら、クライアントはサーバーに対して認証され、サーバーもクライアントに対して認証されたことになります。この時点で信頼が確立され、クライアントとサーバー間でセキュアな通信が行えます。

注記

承認されていないプリンシパルは TGT を使用できませんが、最初に AS で認証に成功した後でのみプリンシパルに TGT が発行されます。これにより、適切に承認されたプリンシパルのみに TGT が発行されるだけでなく、承認されていない第三者がオフラインの辞書攻撃や総当り攻撃などの不正使用が目的で、TGT を取得する可能性が低減されます。

1.5.2. Kerberos および JBoss EAP

JBoss EAP を既存の Kerberos デスクトップベースの SSO 環境と統合すると、同じチケットで JBoss EAP インスタンス上でホストされる web アプリケーションへのアクセスを提供することが可能です。典型的なセットアップでは、レガシー **security** サブシステムまたは **elytron** サブシステムのいずれかを使用して、SPNEGO での Kerberos 認証を使用するよう JBoss EAP インスタンスが設定されます。SPNEGO 認証を使用するよう設定されたアプリケーションは、その JBoss EAP インスタンスにデプロイされます。ユーザーは、Kerberos によって保護されたデスクトップにログインし、KDC と認証の交換を完了します。その後ユーザーは、JBoss EAP インスタンスが直接 web ブラウザーを使用するデプロイされたアプリケーションのセキュアなリソースにアクセスしようとします。JBoss EAP は、セキュアなリソースへのアクセスには承認が必要であることを伝えます。web ブラウザーはユーザーの TGT チケットを取得し、チケットの許可 (承認) を行い、KDC と交換してユーザーを検証し、サービスチケットを取得します。ST がブラウザーに返されたら、SPNEGO 用にフォーマットされたリクエストに ST をラップし、JBoss EAP 上で実行している web アプリケーションに返信します。その後、JBoss EAP は SPNEGO リクエストをアンパックし、レガシー **security** サブシステムまたは **elytron** サブシステムのいずれかを使用して認証を実行します。認証に成功すると、ユーザーはセキュアなリソースへのアクセスが許可されます。

第2章 JBOSS EAP における KERBEROS での SSO のセットアップ方法

2.1. 必要なコンポーネント

JBoss EAP に Kerberos による SSO を設定する場合、以下のコンポーネントが必要になります。

- 適切に設定された Kerberos 環境
- JBoss EAP インスタンス
- web アプリケーション

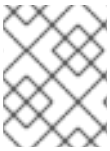
2.1.1. JBoss Negotiation Toolkit

JBoss Negotiation Toolkit は、アプリケーションを本番環境に導入する前に認証メカニズムのデバッグおよびテストを行えるようにするデバッグツールです。これはサポート対象のツールではありませんが、SPENEGO を web アプリケーションに対して設定するのは難しいこともあるため、大変便利なツールと言えます。

JBoss Negotiation Toolkit の事前ビルドされた WAR ファイルは JBoss Negotiation Toolkit リポジトリー からダウンロードできます。JBoss EAP に含まれる JBoss Negotiation のバージョンと一致する JBoss Negotiation Toolkit のバージョンをダウンロードする必要があります。たとえば、JBoss EAP 7.1 をご使用の場合は、バージョンが **3.0.4.Final-redhat-1** の JBoss Negotiation を使用するため、**jboss-negotiation-toolkit-3.0.4.Final.war** を使用する必要があります。使用している JBoss Negotiation のバージョンを確認するには、**EAP_HOME/modules/system/layers/base/org/jboss/security/negotiation/main/module.xml** を確認します。

2.2. KERBEROS 環境

JBoss EAP での Kerberos による SSO の提供 で説明したとおり、Kerberos はサードパーティーの KDC に依存して認証および承認決定を提供します。これには、KDC との認証にブラウザーなどのクライアントとそれらのホストが適切に設定されている必要もあります。本書では主に JBoss EAP とホストされる web アプリケーションを中心に取り上げるため、KDC および Kerberos ドメインの設定については本書の範囲外となります。



注記

これ以降の項では、KDC と Kerberos ドメインがすでにセットアップされ、適切に設定されていることを仮定します。

2.3. これまでのバージョンの JBOSS EAP とは異なる設定

JBoss EAP 7.2 以降と、これまでのバージョンには顕著な違いがいくつかあります。

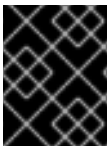
- **jboss-web.xml** には NegotiationAuthenticator バルブが必要でなくなりましたが、引き続き **<security-constraint>** および **<login-config>** 要素を **web.xml** に定義する必要があります。これらの要素は、セキュアなリソースを決定するのに使用されます。
- **<login-config>** 要素の **auth-method** 要素はコンマ区切りリストになりました。**正確な**値の **SPNEGO** が必要で、このリストの 1 番目に記載する必要があります。フォールバックに **FORM** 認証を希望する場合は、**性格**値は **SPNEGO,FORM** になります。

- **elytron** サブシステムを使用する場合は **jboss-deployment-structure.xml** ファイルは必要ありません。

2.4. JBOSS EAP インスタンスの設定

JBoss EAP にデプロイされたアプリケーションが **elytron** または レガシー **security** サブシステムのいずれかを使用するように設定することができますが、両方を使用するように設定することはできません。

2.4.1. Elytron サブシステムの設定



重要

以下の手順は、KDC、Kerberos ドメイン、およびブラウザーが設定され、動作していることを仮定しています。

1. **kerberos-security-factory** を設定します。

```
/subsystem=elytron/kerberos-security-factory=krbSF:add(principal="HTTP/host@REALM",
path="/path/to/http.keytab", mechanism-oids=[1.2.840.113554.1.2.2, 1.3.6.1.5.5.2])
```

2. Kerberos のシステムプロパティーを設定します。
環境の設定に応じて、以下のシステムプロパティーを設定する必要があります。

システムプロパティー	説明
java.security.krb5.kdc	KDC のホスト名。
java.security.krb5.realm	レルムの名前。
java.security.krb5.conf	設定 **krb5.conf** ファイルへのパス。
sun.security.krb5.debug	**true** の場合、デバッグモードが有効になります。

管理 CLI を使用する場合、以下のように JBoss EAP のシステムプロパティーを設定します。

```
/system-property=java.security.krb5.conf:add(value="/path/to/krb5.conf")
```

3. ロールを割り当てるために Elytron セキュリティーレルムを設定します。
クライアントの Kerberos トークンはプリンシパルを提供しますが、そのプリンシパルをロールにマップする方法が必要になります。これには複数の方法がありますが、この例では **filesystem-realm** を作成し、Kerberos トークンからのプリンシパルと一致するレルムにユーザーを追加し、ロールをそのユーザーに割り当てます。

```
/subsystem=elytron/filesystem-realm=exampleFsRealm:add(path=fs-realm-users, relative-
to=jboss.server.config.dir)

/subsystem=elytron/filesystem-realm=exampleFsRealm:add-identity(identity=user1@REALM)
```

```
/subsystem=elytron/filesystem-realm=exampleFsRealm:add-identity-
attribute(identity=user1@REALM, name=Roles, value=["Admin","Guest"])
```

1. **simple-role-decoder** を追加します。

```
/subsystem=elytron/simple-role-decoder=from-roles-attribute:add(attribute=Roles)
```

この **simple-role-decoder** は、**Roles** 属性からのプリンシパルのロールをデコードします。ロールが別の属性にある場合はこの値を変更できます。

2. **security-domain** を設定します。

```
/subsystem=elytron/security-domain=exampleFsSD:add(realms=[{realm=exampleFsRealm,
role-decoder=from-roles-attribute}], default-realm=exampleFsRealm, permission-
mapper=default-permission-mapper)
```

3. **kerberos-security-factory** を使用する **http-authentication-factory** を設定します。

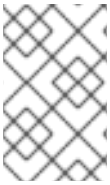
```
/subsystem=elytron/http-authentication-factory=example-krb-http-auth:add(http-server-
mechanism-factory=global, security-domain=exampleFsSD, mechanism-configurations=
[{mechanism-name=SPNEGO, mechanism-realm-configurations=[{realm-
name=exampleFsSD}], credential-security-factory=krbSF}])
```

4. **undertow** サブシステムで **application-security-domain** を設定します。

```
/subsystem=undertow/application-security-domain=app-spnego:add(http-authentication-
factory=example-krb-http-auth)
```

2.4.2. レガシー security サブシステムの設定

JBoss EAP には、デプロイされたアプリケーションと SSO に SPNEGO および JBoss Negotiation を使用して Kerberos を使用するのに必要なすべてのコンポーネントが含まれていますが、以下の設定を変更する必要があります。



注記

ここで使用する管理 CLI コマンドは、JBoss EAP スタンドアロンサーバーを実行していることを仮定しています。JBoss EAP 管理対象ドメインの管理 CLI を使用する場合の詳細は **管理 CLI ガイド** を参照してください。

1. サーバーアイデンティティーまたはホスト、セキュリティードメインの設定
このセキュリティードメインは、コンテナー自体を KDC へ認証します。ユーザーではなくコンテナー自体の認証であるため、静的ログインメカニズムを受容するログインモジュールを使用する必要があります。この例では静的プリンシパルを使用し、クレデンシャルが含まれるキータブファイルを参照します。

例: サーバーアイデンティティーセキュリティードメインの作成

```
/subsystem=security/security-domain=host:add(cache-type=default)

/subsystem=security/security-domain=host/authentication=classic:add()
```

```
/subsystem=security/security-domain=host/authentication=classic/login-
module=Kerberos:add(code=Kerberos, flag=required, module-options=[storeKey=true,
refreshKrb5Config=true, useKeyTab=true, principal=host/testserver@MY_REALM,
keyTab=/home/username/service.keytab, doNotPrompt=true, debug=false])

reload
```

IBM JDK を使用している場合、Kerberos モジュールのオプションは異なります。**jboss.security.disable.secdomain.option** システムプロパティーは **true** に設定する必要があります。詳細は、関連するシステムプロパティーの設定 を参照してください。ログインモジュールは次のように設定する必要があります。

例: IBM JDK

```
/subsystem=security/security-domain=host:add(cache-type=default)

/subsystem=security/security-domain=host/authentication=classic:add()

/subsystem=security/security-domain=host/authentication=classic/login-
module=Kerberos:add(code=Kerberos, flag=required, module-options=
[principal=host/testserver@MY_REALM, keyTab="file:///root/keytab", credsType=acceptor])

reload
```

Kerberos ログインモジュールの設定オプションの完全リストは、JBoss EAPLogin Module Reference を参照してください。

2. web アプリケーションセキュリティードメインを設定します。
web アプリケーションセキュリティードメインは、個別のユーザーを KDC に対して認証するために使用されます。ユーザーの認証には最低でも 1 つのログインモジュールが必要になります。また、ユーザーに適用するロールを検索する方法も必要になります。これは、手動でユーザーをロールにマップする **<mapping>** を追加したり、ユーザーをロールにマップする 2 つ目のログインモジュールを追加するなど、複数の方法で実現できます。

以下は、web アプリケーションセキュリティードメインの例になります。

例: サーバーアイデンティティーセキュリティードメインの作成

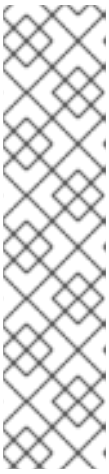
```
/subsystem=security/security-domain=app-spnego:add(cache-type=default)

/subsystem=security/security-domain=app-spnego/authentication=classic:add()

/subsystem=security/security-domain=app-spnego/authentication=classic/login-
module=SPNEGO:add(code=SPNEGO, flag=required, module-options=
[serverSecurityDomain=host])

reload
```

SPNEGO ログインモジュールの設定オプションの完全リストは、JBoss EAPLogin Module Reference を参照してください。

3. 関連するシステムプロパティーの設定
JBoss EAP は、Kerberos サーバーへの接続に関連するシステムプロパティーを設定する機能を提供します。KDC、Kerberos ドメイン、および ネットワーク設定に応じて、以下のシステムプロパティーが必要 (または不必要) になります。

```
<system-properties>
  <property name="java.security.krb5.kdc" value="mykdc.mydomain"/>
  <property name="java.security.krb5.realm" value="MY_REALM"/>
  <property name="java.security.krb5.conf" value="/path/to/krb5.conf"/>
  <property name="jboss.security.disable.secdomain.option" value="true"/>
  <property name="sun.security.krb5.debug" value="false"/>
</system-properties>
```

プロパティー	説明
java.security.krb5.kdc	KDC のホスト名。
java.security.krb5.realm	レルムの名前。
java.security.krb5.conf	設定 **krb5.conf** ファイルへのパス。
jboss.security.disable.secdomain.option	**true** に設定すると、**jboss.security.security_domain** ログインモジュールオプションのセキュリティードメインで宣言されたログイモジュールへの自動追加が無効になります。IBM JDK を使用する場合は **true** に設定する必要があります。
sun.security.krb5.debug	**true** の場合、デバッグモードが有効になります。



注記

デフォルトでは、セキュリティードメインに定義された各ログインモジュールに自動的に **jboss.security.security_domain** モジュールオプションが追加されている必要があります。このオプションは、既知のオプションのみが定義されるようにチェックを行うログインモジュールでは問題が発生します。IBM Kerberos ログインモジュールである **com.ibm.security.auth.module.Krb5LoginModule** はこのようなログインモジュールの 1 つです。このモジュールオプションを追加する動作を無効にするには、JBoss EAP の起動時に **jboss.security.disable.secdomain.option** システムプロパティーを **true** に設定します。これを行うには、管理 CLI または管理コンソールを使用して **<system-properties>** を設定するか、**-Djboss.security.disable.secdomain.option=true** を起動パラメーターに追加します。

システムプロパティーの設定に関する詳細は、JBoss EAP**管理 CLI ガイド** を参照してください。

2.5. WEB アプリケーションの設定

セキュリティードメインが設定されたら、Kerberos 認証を有効にするために、設定したセキュリティードメインを使用するよう web アプリケーションを設定する必要があります。アプリケーションを変更した後、JBoss EAP インスタンスにデプロイして認証に Kerberos を使用することができます。

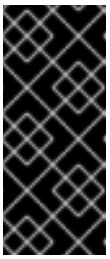
アプリケーションに以下の更新を行う必要があります。

1. SPNEGO 認証メソッドを使用するよう **web.xml** を設定します。
web.xml ファイルには以下が含まれる必要があります。

   - セキュアな領域の URL パターンにマップする **<url-pattern>** が含まれる **<web-resource-collection>** を持つ **<security-constraint>**。任意で、**<security-constraint>** に許可されるロールを明記する **<auth-constraint>** を含めることもできます。

   - **<auth-constraint>** にロールが指定されている場合、これらのロールを **<security-role>** で定義する必要があります。

   - **SPNEGO** の**正確な** 値を持つ **<auth-method>** が含まれる **<login-config>**。



重要

<auth-method> 要素は特定の値のコンマ区切りリストを想定します。**SPNEGO** 認証を適切に設定するには、**正確な**値の **SPNEGO** が最初に **<auth-method>** 要素に記載される必要があります。追加の認証タイプの取り入れについては FORM ログインをフォールバックとして追加 を参照してください。

<security-constraint> および **<security-role>** 要素を使用すると、管理者は URL パターンおよびロールを基に制限された領域または無制限の領域をセットアップできます。これにより、リソースをセキュリティーで保護することができ、保護しないこともできます。

例: web.xml ファイル

```
<web-app>
  <display-name>App1</display-name>
  <description>App1</description>
  <!-- Define a security constraint that requires the Admin role to access resources -->
  <security-constraint>
    <display-name>Security Constraint on Conversation</display-name>
    <web-resource-collection>
      <web-resource-name>exampleWebApp</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>Admin</role-name>
    </auth-constraint>
  </security-constraint>
  <!-- Define the Login Configuration for this Application -->
  <login-config>
    <auth-method>SPNEGO</auth-method>
    <realm-name>SPNEGO</realm-name>
  </login-config>
  <!-- Security roles referenced by this web application -->
  <security-role>
    <description>Role required to log in to the Application</description>
    <role-name>Admin</role-name>
  </security-role>
</web-app>
```

2. 設定されたセキュリティードメインを使用するよう **jboss-web.xml** を設定します。
jboss-web.xml ファイルには以下が必要です。

- 認証または承認に使用されるセキュリティードメインを指定する **<security-domain>**。
- 複数のロール名と照合するために **web.xml** の role-name 要素でアスタリスクの使用を有効にする **<jacc-star-role-allow>** (任意)。

例: jboss-web.xml ファイル

```
<jboss-web>
  <security-domain>app-spnego</security-domain>
  <jacc-star-role-allow>true</jacc-star-role-allow>
</jboss-web>
```

3. JBoss Negotiation の依存関係をレガシー **security** サブシステムのデプロイメントに追加します。

重要

elytron サブシステムを使用している場合は、この手順を省略できます。

SPNEGO および JBoss Negotiation を使用する web アプリケーションでは、JBoss Negotiation クラスが見つかるようにするため、依存関係を **jboss-deployment-structure.xml** に定義する必要があります。JBoss EAP は必要なすべての JBoss Negotiation と関連クラスを提供するため、アプリケーションはこれらの依存関係を宣言して使用することのみが必要となります。

jboss-deployment-structure.xml を使用した依存関係の宣言

```
<jboss-deployment-structure>
  <deployment>
    <dependencies>
      <module name="org.jboss.security.negotiation"/>
    </dependencies>
  </deployment>
</jboss-deployment-structure>
```

この代わりに、依存関係を **META-INF/MANIFEST.MF** ファイルに定義することもできます。

META-INF/MANIFEST.MF を使用して依存関係の宣言

```
Manifest-Version: 1.0
Dependencies: org.jboss.security.negotiation
```

2.6. ACTIVE DIRECTORY に関する注意点

本項では、Active Directory ドメインの一部である Microsoft Windows サーバー上で JBoss EAP が実行されている場合に、SPNEGO 認証の使用で必要となるアカウントの設定方法を説明します。

ここでは、サーバーへのアクセスに使用されるホスト名は **HOST_NAME**、レルムは **REALM**、ドメインは **DOMAIN**、JBoss EAP インスタンスをホストするサーバーは **MACHINE_NAME** を使用します。

2.6.1. Microsoft Windows ドメインの設定

1. 既存のサービスプリンシパルマッピングを消去します。

Microsoft Windows ネットワークでは、一部のマッピングが自動作成されます。自動的に作成されたマッピングを削除し、ネゴシエーションが適切に行われるようにサーバーのアイデンティティーをサービスプリンシパルへマップします。マッピングにより、クライアントコンピューター上の Web ブラウザーがサーバーを信頼し、SPNEGO の実行を試みます。クライアントコンピューターは、HTTP/**HOST_NAME** 形式のマッピングに対し、ドメインコントローラーを検証します。

既存のマッピングを削除する手順は次のとおりです。

以下のコマンドを使用して、コンピューターに対してドメインに登録されたマッピングをリストします。

```
setspn -L MACHINE_NAME
```

以下のコマンドを使用して、既存のマッピングを削除します。

```
setspn -D HTTP/HOST_NAME MACHINE_NAME
```

```
setspn -D host/HOST_NAME MACHINE_NAME
```

2. ホストユーザーアカウントを作成します。

注記

ホスト名には **MACHINE_NAME** 以外の名前を使用してください。

これ以降では、ホストユーザー名として **USER_NAME** を使用します。

3. **USER_NAME** と **HOST_NAME** との間のマッピングを定義します。
以下のコマンドを実行して、サービスプリンシパルマッピングを設定します。

```
ktpass -princ HTTP/HOST_NAME@REALM -pass * [-kvno 0] -mapuser
DOMAIN\USER_NAME -out jboss.keytab -ptype KRB5_NT_PRINCIPAL -crypto all
```

入力を要求されたら、ユーザー名のパスワードを入力します。

コマンド **setspn -L USER_NAME** を実行してマッピングを検証します。

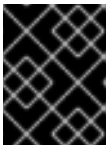
注記

JRE から **KrbException: Specified version of key is not available** エラーを取得した場合、**-kvno 0** を指定してキーバージョンを **0** に設定する必要があることがあります。**REALM** はすべて大文字にする必要がありますが、**HOST_NAME** はすべて小文字にします。また、**HOST_NAME** は FQDN (完全修飾ドメイン名) である必要があり、**CNAME** レコードではなく、解決可能な **A** または **AAAA** レコードである必要があります。

-crypto all の使用は Windows Server 2008 およびそれ以降でのみ動作します。Windows Server 2003 では正確な設定を指定する必要があります。

4. セキュリティードメイン内でプリンシパルを定義します。

プリンシパルは **elytron** または **security** サブシステムで **HTTP/HOST_NAME@REALM** に定義または更新できます。



重要

オプションやパスワードの変更など、ユーザーに変更を加える場合は、**keytab** を再生成する必要があります。

第3章 その他の機能

3.1. FORM ログインをフォールバックとして追加

JBoss EAP および JBoss EAP にデプロイされたアプリケーションでは、FORM ログイン認証メカニズムを設定してフォールバックとして使用することもできます。これにより、アプリケーションは Kerberos/SPNEGO トークンが存在しない場合の認証でログインページを提示できます。この認証は、Kerberos 認証とは独立して行われます。そのため、FORM ログインフォールバックの設定方法によっては、認証に個別のクレデンシャルが必要なことがあります。

注記

SPNEGO または NTLM トークンが存在しない場合、または存在する SPNEGO トークンが他の KDC からである場合に、FORM ログインへのフォールバックが利用できます。

3.1.1. アプリケーションの更新

FORM ログインをフォールバックとして使用するには、アプリケーションの設定に以下の手順が必要になります。

1. Kerberos および SPNEGO を使用するよう JBoss EAP および web アプリケーションを設定します。
   認証および承認に Kerberos および SPNEGO を使用するための、JBoss EAP および web アプリケーションの設定手順は、JBoss EAP における Kerberos での SSO のセットアップ方法 を参照してください。

2. ログインおよびエラーページを追加します。
   FORM ログインを使用するには、ログインおよびエラーページが必要です。これらのファイルは web アプリケーションに追加され、認証プロセスで使用されます。

   例: login.jsp ファイル

```
<html>
  <head></head>
  <body>
    <form id="login_form" name="login_form" method="post" action="j_security_check"
enctype="application/x-www-form-urlencoded">
      <center> <p>Please login to proceed.</p> </center>
      <div style="margin-left: 15px;">
        <p> <label for="username">Username</label> <br /> <input id="username" type="text"
name="j_username"/> </p>
        <p> <label for="password">Password</label> <br /> <input id="password"
type="password" name="j_password" value=""/> </p>
        <center> <input id="submit" type="submit" name="submit" value="Login"/> </center>
      </div>
    </form>
  </body>
</html>
```

   例: error.jsp ファイル

```
<html>
  <head></head>
```

```
  <body>
    <p>Login failed, please go back and try again.</p>
  </body>
</html>
```

3. **web.xml** を編集します。
ログインおよびエラーページを web アプリケーションに追加した後、**web.xml** を更新して FORM ログインでこれらのファイルが使用されるようにする必要があります。**FORM** という**正確な**値をそのまま **<auth-method>** 要素に追加する必要があります。**<auth-method>** はコンマ区切りリストを想定し、順番が重要であるため、**<auth-method>** の値を**正確**に **SPNEGO,FORM** に更新する必要があります。さらに、**<form-login-config>** 要素を **<login-config>** と、**<form-login-page>** および **<form-error-page>** 要素として指定されたログインおよびエラーページへのパスに追加する必要があります。

例: 更新された web.xml ファイル

```
<web-app>
  <display-name>App1</display-name>
  <description>App1</description>
  <!-- Define a security constraint that requires the Admin role to access resources -->
  <security-constraint>
    <display-name>Security Constraint on Conversation</display-name>
    <web-resource-collection>
      <web-resource-name>examplesWebApp</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
       <role-name>Admin</role-name>
    </auth-constraint>
  </security-constraint>
  <!-- Define the Login Configuration for this Application -->
  <login-config>
    <auth-method>SPNEGO,FORM</auth-method>
    <realm-name>SPNEGO</realm-name>
    <form-login-config>
      <form-login-page>/login.jsp</form-login-page>
      <form-error-page>/error.jsp</form-error-page>
    </form-login-config>
  </login-config>
  <!-- Security roles referenced by this web application -->
  <security-role>
    <description> role required to log in to the Application</description>
    <role-name>Admin</role-name>
  </security-role>
</web-app>
```

3.1.2. Elytron サブシステムの更新

1. **http-authentication-factory** に **FORM** 認証のメカニズムを追加します。
Kerberos ベースの認証向けに設定した既存の **http-authentication-factory** と、**FORM** 認証の追加のメカニズムを使用できます。

```
/subsystem=elytron/http-authentication-factory=example-krb-http-auth:list-
add(name=mechanism-configurations, value={mechanism-name=FORM})
```

2. フォールバックプリンシパルを追加します。
   Kerberos ベースの認証向けの既存の設定には、プリンシパルを Kerberos トークンからアプリケーションのロールへマッピングするために設定されたセキュリティーレルムがすでに含まれているはずです。フォールバック認証向けの追加ユーザーをそのレルムに追加できます。たとえば、**filesystem-realm** を使用する場合、適切なロールを持つ新規ユーザーを作成できます。

```
/subsystem=elytron/filesystem-realm=exampleFsRealm:add-identity(identity=fallbackUser1)

/subsystem=elytron/filesystem-realm=exampleFsRealm:set-password(identity=fallbackUser1, clear={password="password123"})

/subsystem=elytron/filesystem-realm=exampleFsRealm:add-identity-attribute(identity=fallbackUser1, name=Roles, value=["Admin","Guest"])
```

3.1.3. レガシー Security サブシステムの更新

JBoss EAP でレガシー **security** サウシステムを使用している場合は、フォールバック認証のセキュリティードメインを更新する必要があります。

web アプリケーションセキュリティードメインを設定して、フォールバックログインメカニズムをサポートする必要があります。これには以下の手順が必要になります。

1. フォールバック認証メソッドとして対応する新しいセキュリティードメインを追加します。
2. **usernamePasswordDomain** モジュールオプションを、フォールバックドメインを示す web アプリケーションセキュリティードメインに追加します。

例: フォールバックセキュリティードメインで設定されたセキュリティードメイン

```
/subsystem=security/security-domain=app-fallback:add(cache-type=default)

/subsystem=security/security-domain=app-fallback/authentication=classic:add()

/subsystem=security/security-domain=app-fallback/authentication=classic/login-module=UsersRoles:add(code=UsersRoles, flag=required, module-options=[usersProperties="file:${jboss.server.config.dir}/fallback-users.properties", rolesProperties="file:${jboss.server.config.dir}/fallback-roles.properties"])

/subsystem=security/security-domain=app-spnego/authentication=classic/login-module=SPNEGO:add(code=SPNEGO, flag=required, module-options=[serverSecurityDomain=host])

/subsystem=security/security-domain=app-spnego/authentication=classic/login-module=SPNEGO:map-put(name=module-options, key=usernamePasswordDomain, value=app-fallback)

/subsystem=security/security-domain=app-spnego/authentication=classic/login-module=SPNEGO:map-put(name=module-options, key=password-stacking, value=useFirstPass)

reload
```

3.2. KERBEROS での管理インターフェイスのセキュア化

JBoss EAP はセキュリティードメインで Kerberos 認証を提供する他に、Kerberos を使用して管理インターフェイスをセキュアにする機能も提供します。

3.2.1. Elytron を使用した Kerberos での管理インターフェイスのセキュア化

HTTP 管理インターフェイスに Kerberos 認証を設定するには、以下を行います。

1. アプリケーションに Kerberos 認証を設定する手順 にしたがって、Kerberos 認証を行う **http-authentication-factory** を作成します。

重要

管理インターフェイスに Kerberos 認証を設定するとき、JBoss EAP が KDC に対して認証行うために設定したサービスプリンシパルに注意することが大変重要になります。このサービスプリンシパルは **service-name/hostname** の形式を取ります。JBoss EAP は、web ベースの管理コンソールに対して認証を行うときは **HTTP/localhost** のように **HTTP** がサービス名になることを想定し、管理 CLI の場合は **remote/localhost** のように **remote** がサービス名になること想定します。

2. **http-authentication-factory** を使用するよう、管理 HTTP インターフェイスを更新します。

```
/core-service=management/management-interface=http-interface:write-attribute(name=http-authentication-factory, value=example-krb-http-auth)
```

管理 CLI で SASL 認証に対する Kerberos 認証を設定するには、以下を行います。

1. アプリケーションに Kerberos 認証を設定する手順 にしたがって、セキュリティードメインおよび **kerberos-security-factory** を作成します。

2. **GSSAPI** を **configurable-sasl-server-factory** に追加します。

```
/subsystem=elytron/configurable-sasl-server-factory=configured:list-add(name=filters, value={pattern-filter=GSSAPI})
```

3. セキュリティードメインおよび **kerberos-security-factory** を使用する **sasl-authentication-factory** を作成します。

例: sasl-authentication-factory

```
/subsystem=elytron/sasl-authentication-factory=example-sasl-auth:add(sasl-server-factory=configured, security-domain=exampleFsSD, mechanism-configurations=[{mechanism-name=GSSAPI, mechanism-realm-configurations=[{realm-name=exampleFsSD}], credential-security-factory=krbSF}])
```

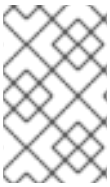
4. **sasl-authentication-factory** を使用するよう、管理 SASL インターフェイスを更新します。

例: sasl-authentication-factory の更新

```
/core-service=management/management-interface=http-interface:write-attribute(name=http-upgrade.sasl-authentication-factory, value=example-sasl-auth)

reload
```

3.2.2. レガシーのコア管理認証を使用した Kerberos での管理インターフェイスのセキュア化

レガシーのコア管理認証を使用して管理インターフェイスでの Kerberos 認証を有効にするには、以下の手順を実行する必要があります。



注記

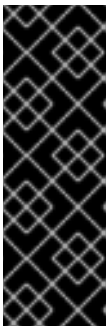
ここで使用する管理 CLI コマンドは、JBoss EAP スタンドアロンサーバーを実行していることを仮定しています。JBoss EAP 管理対象ドメインの管理 CLI を使用する場合の詳細は **管理 CLI ガイド** を参照してください。

1. 関連するシステムプロパティーを有効にします。
   前の項 で説明したとおり、Kerberos サーバーへの接続に必要な JBoss EAP システムプロパティーを有効にします。

2. Kerberos サーバーアイデンティティーをセキュリティーレルムに追加します。
   Kerberos 認証をセキュリティーレルムで使用できるようにするには、Kerberos サーバーへの接続を追加する必要があります。以下の例は、Kerberos サーバーアイデンティティーを既存の管理レルムに追加する方法を表しています。**service-name**、**hostname**、および **MY-REALM** は適切な値に置き換える必要があります。

   サーバーアイデンティティーをセキュリティーレルムに追加する CLI の例

   ```
   /core-service=management/security-realm=ManagementRealm/server-identity=kerberos:add()

   /core-service=management/security-realm=ManagementRealm/server-identity=kerberos/keytab=service-name\/hostname@MY-REALM:add(path=/home\/username\/service.keytab, debug=true)

   reload
   ```

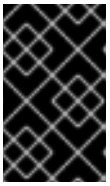
   重要

   管理インターフェイスに Kerberos 認証を設定するとき、JBoss EAP が KDC に対して認証行うために設定したサービスプリンシパルに注意することが大変重要になります。このサービスプリンシパルは **service-name/hostname** の形式を取ります。JBoss EAP は、web ベースの管理コンソールに対して認証を行うときは **HTTP/localhost** のように **HTTP** がサービス名になることを想定し、管理 CLI の場合は **remote/localhost** のように **remote** がサービス名になること想定します。

3. セキュリティーレルムで認証方法を更新します。
   Kerberos サーバーアイデンティティーが適切に設定された後、使用するためにはセキュリティーレルム認証メソッドを更新する必要があります。

   例: Kerberos 認証のセキュリティーレルムへの追加

   ```
   /core-service=management/security-realm=ManagementRealm/authentication=kerberos:add()

   reload
   ```



重要

管理インターフェイスにアクセスするとき、JBoss EAP はセキュリティーレルムで定義された認証メカニズムの順番を基に、その順番でユーザーを認証しようとします。

4. 両方のインターフェイスを Kerberos でセキュアにします。
web ベースの管理コンソールと管理 CLI の両方をセキュアする場合、 Kerberos サーバーアイデンティティーを両方に設定する必要があります。アイデンティティーを追加するには、以下のコマンドを使用します。

```
/core-service=management/security-realm=ManagementRealm/server-identity=kerberos/keytab=remote\/hostname@MY-REALM:add(path=/home\/username\/remote.keytab, debug=true)

reload
```

3.2.3. 管理インターフェイスへの接続

管理インターフェイスに接続する前に、有効な Kerberos チケットが必要になります。レガシーのセキュリティーソリューションを使用して、セキュリティーレルムが Kerberos によるユーザーの認証に失敗すると、**<authentication>** 要素に指定されたその他の方法を使用してユーザーの認証を試みます。**elytron** サブシステムはレガシーのセキュリティーソリューションと同様に動作します。Kerberos 認証メカニズムに失敗すると、認証は管理インターフェイスを保護する認証ファクトリーに定義したその他のメカニズムにフォールバックします。通常、DIGEST または BASIC がフォールバックとして使用されます。

ブラウザーを使用して web ベースの管理コンソールに接続する場合、セキュリティーレルムはそのチケットを基にユーザーを認証しようとします。

管理 CLI に接続する場合、GSSAPI 実装がオペレーティングシステムのレベルで管理されたアイデンティティーを利用できるようにするため、 **-Djavax.security.auth.useSubjectCredsOnly=false** パラメーターを使用する必要があります。環境のセットアップを基にして以下のパラメーターを使用する必要もあります。

-Djava.security.krb5.realm=REALM_NAME

レルム名を指定します。

-Djava.security.krb5.kdc=KDC_HOSTNAME

KDC の場所を指定します。

--no-local-auth

ローカル認証を無効にします。これは、スクリプトを実行している同じマシン上で JBoss EAP インスタンスへの接続を試みる場合に便利です。

コマンド例

```
$ EAP_HOME/bin/jboss-cli.sh -c -Djavax.security.auth.useSubjectCredsOnly=false --no-local-auth
```

警告

クライアントとサーバー間で HTTP プロキシーが使用される場合、別の認証されたクライアントと同じサーバー間で認証された接続を共有しないように注意する必要があります。これを怠った場合、サーバーはセキュリティーコンテキストの関連を簡単に認識できないようになります。クライアントとサーバー間における認証の整合性を適切に維持しているプロキシーは、プロキシーからの HTTP 応答で **Proxy-support: Session- Based-Authentication** HTTP ヘッダーをクライアントに提供します。プロキシーがこのヘッダーとサーバーからの **401 Unauthorized** 応答を提供する場合を除き、クライアントはプロキシーを介して SPNEGO HTTP 認証メカニズムを利用しては **なりません**。

3.3. リモーティングの KERBEROS 認証統合

Kerberos を使用して管理インターフェイスと web アプリケーションをセキュアにする他に、EJB などのリモーティングでアクセスされるサービスに対して Kerberos 認証を設定することもできます。

Kerberos のシステムプロパティーを設定する必要もあります。詳細は Elytron サブシステムの設定 を参照してください。

3.3.1. レガシーセキュリティーレルムを使用した Kerberos 認証の統合

Kerberos 認証を設定するには、以下を行う必要があります。

1. リモーティングおよび **RealmDirect** でのセキュリティードメインの設定
   リモーティングによってアクセスされるサービスが使用するセキュリティードメインを設定する必要があります。このセキュリティードメインは、**RealmDirect** や **RealmUsersRoles** などの **Remoting** ログインモジュールと **RealmDirect** ログインモジュールの両方を利用する必要があります。本質的には、デフォルトで提供される **other** セキュリティードメインと大変似ています。各ログインモジュールの設定オプションの詳細は、JBoss EAPLogin Module Reference を参照してください。

   例: Remoting および RealmDirect ログインモジュールが指定されたセキュリティードメイン

```
/subsystem=security/security-domain=krb-remoting-domain:add()

/subsystem=security/security-domain=krb-remoting-domain/authentication=classic:add()

/subsystem=security/security-domain=krb-remoting-domain/authentication=classic/login-module=Remoting:add(code=Remoting, flag=optional, module-options=[password-stacking=useFirstPass])

/subsystem=security/security-domain=krb-remoting-domain/authentication=classic/login-module=RealmDirect:add(code=RealmDirect, flag=required, module-options=[password-stacking=useFirstPass, realm=krbRealm])

/subsystem=security/security-domain=krb-remoting-domain/mapping=classic:add()

/subsystem=security/security-domain=krb-remoting-domain/mapping=classic/mapping-module=SimpleRoles:add(code=SimpleRoles, type=role, module-options=
```

```
["testUser"="testRole"])

reload
```

2. Kerberos 認証のセキュリティーレルムを設定します。
Kerberos 認証によるセキュリティーレルムのセットアップについては、Kerberos での管理インターフェイスのセキュア化 を参照してください。

例: セキュリティーレルム

```
/core-service=management/security-realm=krbRealm:add()

/core-service=management/security-realm=krbRealm/server-identity=kerberos:add()

/core-service=management/security-realm=krbRealm/server-identity=kerberos/keytab=remote\/localhost@JBOSS.ORG:add(path=\/path\/to\/remote.keytab, debug=true)

/core-service=management/security-realm=krbRealm/authentication=kerberos:add(remove-realm=true)

reload
```

3. **remoting** サブシステムに HTTP コネクターを設定します。
さらに、新規作成されたセキュリティーレルムを使用するために **remoting** サブシステムで HTTP コネクターを設定する必要があります。

例: remoting サブシステム

```
/subsystem=remoting/http-connector=http-remoting-connector:write-attribute(name=security-realm, value=krbRealm)
```

4. サービスのセキュリティーを設定します。
secured へのリモーティングインターフェイスを使用してアクセスされるサービスを設定する必要もあります。この設定はサービスによって異なります。たとえば、EJB の場合、**@SecurityDomain** および **@RolesAllowed** アノテーションを使用できます。

3.3.2. Elytron を使用した Kerberos 認証の統合

Kerberos または GSSAPI SASL 認証の Elytron セキュリティードメインをリモーティング認証に対して定義できます。

1. アイデンティティーのロード元となるセキュリティーレルムを定義します。これは、ロールの割り当てに使用されます。

```
/path=kerberos:add(relative-to=user.home, path=src/kerberos)

/subsystem=elytron/properties-realm=kerberos-properties:add(users-properties={path=kerberos-users.properties, relative-to=kerberos, digest-realm-name=ELYTRON.ORG}, groups-properties={path=kerberos-groups.properties, relative-to=kerberos})
```

2. サーバーのアイデンティティーの Kerberos セキュリティーファクトリーを定義します。

```
/subsystem=elytron/kerberos-security-factory=test-server:add(relative-to=kerberos,
path=remote-test-server.keytab, principal=remote/test-server.elytron.org@ELYTRON.ORG)
```

3. セキュリティードメインと SASL 認証ファクトリーを定義します。

```
/subsystem=elytron/security-domain=KerberosDomain:add(default-realm=kerberos-
properties, realms=[{realm=kerberos-properties, role-decoder=groups-to-roles}], permission-
mapper=default-permission-mapper)

/subsystem=elytron/sasl-authentication-factory=gssapi-authentication-factory:add(security-
domain=KerberosDomain, sasl-server-factory=elytron, mechanism-configurations=
[{mechanism-name=GSSAPI, credential-security-factory=test-server}])
```

4. 作成した **sasl-authentication-factory** を **remoting** サブシステムで使用し、リモーティングで有効にします。

   CLI コマンドの例

```
/subsystem=remoting/http-connector=http-remoting-connector:write-attribute(name=sasl-
authentication-factory, value=gssapi-authentication-factory)
```

5. サービスのセキュリティーを設定します。
   EJB でセキュリティードメインを参照する場合は、Elytron セキュリティードメインにマップする **application-security-domain** を指定する必要があります。たとえば EJB の場合では、**@SecurityDomain** アノテーションを使用できます。

   CLI コマンドの例

```
/subsystem=ejb3/application-security-domain=KerberosDomain:add(security-
domain=KerberosDomain)
```

アイデンティティアソシエーションに JAAS Subject は使用できなくなりました。EJB 呼び出しの Kerberos アイデンティティーをプログラミングで管理するクライアントは、以下のように **AuthenticationConfiguration** API を直接使用するよう移行する必要があります。

```
// create your authentication configuration
AuthenticationConfiguration configuration = AuthenticationConfiguration.empty()
    .useProvidersFromClassLoader(SecuredGSSCredentialClient.class.getClassLoader())
    .useGSSCredential(getGSSCredential());

// create your authentication context
AuthenticationContext context = AuthenticationContext.empty().with(MatchRule.ALL, configuration);

// create a callable that looks up an EJB and invokes a method on it
Callable<Void> callable = () -> {
...
};

// use your authentication context to run your callable
context.runCallable(callable);
```

AuthenticationConfiguration の作成時に **useGSSCredential(getGSSCredential())** への呼び出しが発生します。JAAS Subject へすでにアクセス済みのクライアントコードは、以下のように **GSSCredential** を取得するよう簡単に変換できます。

```
private GSSCredential getGSSCredential() {
    return Subject.doAs(subject, new PrivilegedAction<GSSCredential>() {

        public GSSCredential run() {
            try {
                GSSManager gssManager = GSSManager.getInstance();
                return gssManager.createCredential(GSSCredential.INITIATE_ONLY);
            } catch (Exception e) {
                e.printStackTrace();
            }
            return null;
        }
    });
}
```

Revised on 2023-01-28 12:56:49 +1000