Red Hat JBoss Enterprise Application Platform 7.4

如何使用 Kerberos 设置 SSO

使用 Kerberos 配置和管理红帽 JBoss 企业应用平台的单点登录用户访问的说明。

Last Updated: 2024-02-10

Red Hat JBoss Enterprise Application Platform 7.4 如何使用 Kerberos 设置 SSO

使用 Kerberos 配置和管理红帽 JBoss 企业应用平台的单点登录用户访问的说明。

法律通告

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

摘要

本指南旨在探讨红帽 JBoss 企业应用平台内 Kerberos 单点登录(SSO)的主题，并提供在 JBoss EAP 中使用 Kerberos 设置 SSO 的实用指南。本指南将更加深入地探讨使用 Kerberos 的 SSO 以及如何在 JBoss EAP 中进行设置和配置。在阅读本指南前，用户应阅读 Red Hat JBoss Enterprise Application Platform 的安全架构文档，并充分了解本文档中介绍的 SSO 和 Kerberos 信息。 本指南还利用 JBoss EAP CLI 界面执行配置更改。有关将 CLI 用于独立 JBoss EAP 实例和 JBoss EAP 域的更多信息，请参阅 JBoss EAP 管理 CLI 指南。完成本指南时，读者应当对 SSO 和 Kerberos 有了扎实的工作了解、与 JBoss EAP 的关系以及如何进行配置。

目录

提供有关 JBOSS EAP 文档的反馈

要报告错误或改进文档，请登录到 Red Hat JIRA 帐户并提交问题。如果您没有 Red Hat Jira 帐户，则会提示您创建一个帐户。

流程

1. 单击以下链接 **以创建 ticket**。
2. 请包含 **文档 URL、章节编号 并描述问题**。
3. 在 **Summary** 中输入问题的简短描述。
4. 在 **Description** 中提供问题或功能增强的详细描述。包括一个指向文档中问题的 URL。
5. 点 **Submit** 创建问题，并将问题路由到适当的文档团队。

使开源包含更多

红帽承诺替换我们的代码、文档和网页属性中存在问题的语言。我们从这四个术语开始： master、slave、blacklist 和 whitelist。这些更改将在即将发行的几个发行本中逐渐实施。详情请查看 CTO Chris Wright 信息。

第 1 章 带有 KERBEROS DEEPER DIVE 的 SSO

1.1. SSO 和 KERBEROS 是什么？

JBoss EAP 安全架构 *指南* 的单点登录(SSO)和 Kerberos 部分提供了 单点登录 (SSO)和 Kerberos 的基本背景。

1.2. KERBEROS 组件

Kerberos 本身是一种网络协议，通过使用 secret-key 加密为客户端/服务器应用的用户启用身份验证。Kerberos 通常用于对网络上的桌面用户进行身份验证，但通过使用一些其他工具，它可用于对用户进行 Web 应用的身份验证并为一组 Web 应用提供 SSO。这基本上允许已在桌面网络上通过身份验证的用户无缝访问 Web 应用程序中的安全资源，而无需重新身份验证。此概念称为基于桌面的 SSO，因为用户是使用基于桌面的身份验证机制进行身份验证，并且 Web 应用也在使用其身份验证令牌或票据。这与其他 SSO 机制（如基于浏览器的 SSO）不同，后者通过浏览器验证用户和签发令牌。

Kerberos 协议定义了它在身份验证和授权中使用的几个组件：

票证

ticket 是一种安全令牌形式，Kerberos 使用它针对主体发布和身份验证及授权决策。

身份验证服务

身份验证服务(AS)在主体首次登录网络时需要登录。身份验证服务负责签发票据授予票据(TGT)，这是根据票据授予服务和随后访问受保护服务和资源进行验证所需的。

ticket 授予服务

票据授予服务 (TGS)负责向主体和他们尝试访问的目标服务器发出服务票据和特定的会话信息。这基于由主体提供的 TGT 和目的地信息。然后，使用此服务票据和会话信息建立与目的地的连接，并访问所需的安全服务或资源。

密钥分发中心

关键分发中心 (KDC)是同时承载 TGS 和 AS 的组件。KDC 以及客户端或主体以及服务器或安全服务是执行 Kerberos 身份验证所需的三个部分。

ticket 授予票据

授予票据(TGT)是 AS 向主体发出的票据类型。TGT 且通过其用户名和密码针对 AS 成功进行了验证后被授予 TGT。TGT 由客户端在本地缓存，但经过加密后，只有 KDC 可以读取它并且客户端无法读取。这允许 AS 在 TGT 中安全地存储授权数据和其他信息，供 TGS 使用，并让 TGS 能够使用此数据做出授权决策。

Service Ticket

服务票据 (ST)是基于 TGS 的 TGT 和预期目的地向主体发出的票据类型。主体为 TGS 提供 TGT 和预期目的地，TGS 验证主体是否可以根据 TGT 中的授权数据访问目的地。如果成功，TGS 会为客户端和目的地服务器（即含有受保护服务或资源的服务器）向客户端发出 ST。这授予客户端对目标服务器的访问权限。ST 由客户端和服务器缓存，并且可由客户端和服务器读取，也包含允许客户端和服务器安全通信的会话信息。

注意

Kerberos 与网络的 DNS 设置之间有紧密的关系。例如，当客户端根据它所运行的主机的名称访问 KDC 时，会做出某些假设。因此，务必要正确配置除 Kerberos 设置之外的所有 DNS 设置，以确保客户端能够连接。

1.3. 其他组件

除了 Kerberos 组件外，还需要其他几个项目来启用 JBoss EAP 的 Kerberos SSO。

1.3.1. SPNEGO

简单且受保护的 GSSAPI 协商机制 (SPNEGO)提供了一种机制，可用于扩展基于 Kerberos 的单点登录环境，供 Web 应用程序使用。

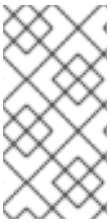
SPNEGO 是客户端应用用于向服务器验证自身的身份验证方法。当客户端应用和服务器尝试互相通信时，可使用这一技术，但是都不确定彼此支持的身份验证协议。SPNEGO 确定客户端应用程序和服务器之间的通用 GSSAPI 机制，然后为其分配其他所有安全操作。

当客户端计算机上的应用（如 Web 浏览器）尝试访问 Web 服务器上的受保护页面时，服务器需要响应该授权。然后，应用程序从 Kerberos KDC 请求服务票据。获取票据后，应用程序将其以格式化为 SPNEGO 的请求打包，并通过浏览器将其发回到 Web 应用。运行部署的 Web 应用的 Web 容器解压缩请求并尝试对票据进行身份验证。身份验证成功后，将授予访问权限。

SPNEGO 与所有类型的 Kerberos 提供商合作，包括红帽企业 Linux 中包含的 Kerberos 服务和 Kerberos 服务器（它们是 Microsoft Active Directory 不可或缺的组成部分）。

1.3.2. JBoss Negotiation

JBoss Negotiation 是 JBoss EAP 附带的一个框架，提供身份验证程序和 Jakarta 身份验证登录模块，以支持 JBoss EAP 中的 SPNEGO。JBoss 协商仅与传统 **安全** 子系统和旧版核心管理身份验证一起使用。有关 Jakarta 身份验证登录模块的更多信息，请参阅 JBoss EAP 安全架构 *指南中 的声明安全性和 安全域* 章节。



注意

使用 JBoss Negotiation 保护某些应用（如 REST Web 服务）时，可能会在超时时间内创建一个或多个会话并保持打开状态，当客户端发出请求时，默认为 30 分钟。这与使用基本身份验证保护应用的预期行为不同，后者不会留下任何打开的会话。实施 JBoss 协商是指使用会话来维护协商/连接的状态，以便创建这些会话是预期的行为。

1.4. KERBEROS 集成

Kerberos 与包括 Red Hat Enterprise Linux 等 Linux 发行版在内的许多操作系统集成。Kerberos 也是 Microsoft Active Directory 的完整组成部分，受到红帽目录服务器和红帽 IDM 的支持。

1.5. KERBEROS 如何为 JBOSS EAP 提供 SSO？

Kerberos 通过向客户端和服务器发出 KDC 票据，提供基于桌面的 SSO。JBoss EAP 可以通过在自己的身份验证和授权流程中使用相同的票据，集成此现有流程。在尝试了解 JBoss EAP 如何重复使用这些票据之前，最好先更加详细地了解发出这些票据的方式，以及不使用 JBoss EAP 的 SSO 与 Kerberos 配合使用的身份验证和授权。

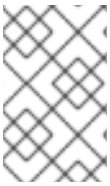
1.5.1. 在基于桌面的 SSO 中使用 Kerberos 进行身份验证和授权

为提供身份验证和授权，Kerberos 依靠第三方 KDC 为访问服务器的客户端提供身份验证和授权决策。这些决定分为三个步骤：

1. 身份验证交换.
   当主体首先访问网络或尝试访问受保护的服务时，无需票据授予票据(TGT)时，他们将面临通过其凭证对身份验证服务(AS)进行身份验证的挑战。AS 根据配置的身份存储验证用户提供的凭据，在成功通过身份验证后，主体将发出 TGT，由客户端缓存。TGT 还包含一些会话信息，从而保证将来客户端和 KDC 之间的通信安全。

2. 票据授予或授权交换.
   委托人获得 TGT 后，即可尝试访问受保护的服务或资源。主体发送请求到向问题单授予服务(TGS)，传递由 KDC 发出的 TGT，并为特定目的地请求服务票据(ST)。TGS 检查由主体提供的 TGT，并验证它们是否具有访问请求的资源的适当权限。如果成功，TGS 会发出 ST 供主体访问该特定目的地。TGS 还为客户端和目标服务器创建会话信息，以允许两者之间的安全通信。此会话信息是单独加密的，这样客户端和服务器只能使用 KDC 单独提供的长期密钥来解密其自身的会话信息，与之前的事务。TGS 随后通过 ST 响应客户端，该 ST 包含客户端和服务器的会话信息。

3. 访问服务器.
   现在主体有安全服务 ST 以及与该服务器进行安全通信的机制，客户端现在可以建立连接并尝试访问受保护的资源。客户端首先将 ST 传递到目标服务器。此 ST 包含从该目的地从 TGS 收到的会话信息的服务器组件。服务器尝试使用来自 KDC 的长期密钥解密客户端传递给它的会话信息。如果成功，客户端已成功通过服务器的身份验证，并且该服务器也被视为对客户端进行身份验证。此时，在客户端和服务器之间建立了信任并可以继续进行通信。



注意

尽管未经授权的主体实际上无法使用 TGT，但只有在他们首次与 AS 成功验证后，才会颁发 TGT。这不仅可确保只签发了经适当授权的主体，而且还会降低未经授权的第三方获取 TGT 的能力，以试图破坏或利用它们，例如使用脱机词典或暴力攻击。

1.5.2. Kerberos 和 JBoss EAP

JBoss EAP 可以集成现有的基于 Kerberos 桌面的 SSO 环境，允许这些相同的票据访问托管在 JBoss EAP 实例上的 Web 应用。在典型的设置中，JBoss EAP 实例将配置为使用传统 **安全** 子系统或 **elytron** 子系统通过 SPNEGO 的 Kerberos 身份验证。配置为使用 SPNEGO 身份验证的应用已部署到 JBoss EAP 实例。用户登录桌面（由 Kerberos 管理），并与 KDC 完成身份验证交换。然后，用户尝试直接使用 Web 浏览器访问部署的应用中的安全资源。JBoss EAP 响应需要授权才能访问受保护的资源。Web 浏览器获取用户的 TGT 票据，然后执行授予或授权与 KDC 交换的票据，以验证用户并获取服务票据。将 ST 返回到浏览器后，它会以 SPNEGO 格式的请求括起 ST，再将它发回到 JBoss EAP 上运行的 Web 应用。JBoss EAP 随后解压缩 SPNEGO 请求，并使用传统 **安全** 子系统或 **elytron** 子系统执行身份验证。如果身份验证成功，则会授予用户对受保护资源的访问权限。

第 2 章 如何使用 KERBEROS 为 JBOSS EAP 设置 SSO

2.1. 所需的组件

在为带有 Kerberos 的 SSO 设置 JBoss EAP 时，您必须具有以下组件：

- 正确配置的 Kerberos 环境
- JBoss EAP 实例
- Web 应用程序

2.1.1. 关于 JBoss 协商工具包

JBoss Negotiation Toolkit 是一个调试工具，可帮助用户在将应用程序引入生产之前调试和测试身份验证机制。它不受支持的工具，但非常有用，因为 SPNEGO 很难为 Web 应用程序配置。

您可以从 JBoss Negotiation Toolkit 存储库 下载 JBoss Negotiation Toolkit 的预构建 WAR 文件。您应该下载与 JBoss EAP 中包括的 JBoss Negotiation 版本相匹配的 JBoss Negotiation Toolkit 版本。例如，如果您正在使用使用 JBoss Negotiation **3.0.4.Final-redhat-1** 的 JBoss EAP 7.1，则应使用 **jboss-negotiation-toolkit-3.0.4.Final.war**。通过查看 ***EAP_HOME*/modules/layers/base/org/jboss/security/negotiation/main/module.xml**，您可以确定正在使用哪个版本的 JBoss Negotiation。

2.2. KERBEROS 环境

如 Kerberos 如何为 JBoss EAP 提供 SSO 中所述， Kerberos 依赖于 KDC 第三方来提供身份验证和授权决策。这还要求客户端（如浏览器），并且正确配置其主机以与 KDC 进行身份验证。本指南主要侧重于如何配置 JBoss EAP 及其托管 Web 应用程序，因此配置 KDC 和 Kerberos 域不在本文的讨论范围之内。

注意

随后部分假设 KDC 和 Kerberos 域已经设置并正确配置。

2.3. 配置旧版本 JBOSS EAP 的不同

JBoss EAP 7.2 或更高版本和更早版本之间有一些明显的区别：

- **jboss-web.xml 中不再需要 NegotiationAuthenticator valve，但 web.xml** 中定义的 **<security-constraint>** 和 **<login-config>** 元素仍需要。它们被用来决定哪些资源受到保护。
- **<login -config> 元素中的 auth-method** 元素现在是一个用逗号分开的列表。**SPNEGO** *的确切*值必须存在，并且应首先显示在该列表中。如果需要将 **FORM** 身份验证作为回退，则 *确切的* 值为 **SPNEGO,FORM**。
- 使用 **elytron** 子系统时，不需要 **jboss-deployment-structure.xml** 文件。

2.4. 配置 JBOSS EAP 实例

您可以将部署到 JBoss EAP 的应用配置为使用 **elytron** 或 传统 **安全** 子系统，但您无法将它配置为使用两者。

2.4.1. 配置 Elytron 子系统

重要

以下步骤假定您已配置且正常工作的 KDC、Kerberos 域和浏览器。

1. 配置 **kerberos-security-factory**。

```
/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. 配置 Kerberos 的系统属性.
根据您的环境配置方式，您需要设置以下部分系统属性。

系统属性	描述
java.security.krb5.kdc	KDC 的主机名。
java.security.krb5.realm	域的名称。
java.security.krb5.conf	配置 **krb5.conf** 文件的路径。
sun.security.krb5.debug	如果为 **true**，则会启用调试模式。

使用管理 CLI 在 JBoss EAP 中配置系统属性：

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

3. 配置 Elytron 安全域以分配角色。
客户端的 Kerberos 令牌将提供主体，但您需要设法将该主体映射到您的应用的角色。实现这一目标的方法有多种，但本例创建了 **文件系统realm**，将用户添加到与 Kerberos 令牌中主体匹配的域中，并将角色分配给该用户。

```
/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. 添加 **simple-role-decoder**。

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

此 **simple-role-decoder** 从 **Roles** 属性解码主体的角色。如果您的角色在不同的属性中，您可以更改此值。

2. 配置 **安全域.**

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

3. 配置 **http-authentication-factory**，它使用 **kerberos-security-factory**。

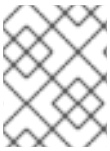
```
/subsystem=elytron/http-authentication-factory=example-krb-http-auth:add(http-server-
mechanism-factory=global, security-domain=exampleFsSD, mechanism-configurations=
[{mechanism-name=SPNEGO, mechanism-realm-configurations=[{realm-
name=exampleFsSD}], credential-security-factory=krbSF}])
```

4. 在 **undertow** 子系统中配置 **application-security-domain** ：

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

2.4.2. 配置传统安全子系统

JBoss EAP 附带了将 Kerberos（使用 SPNEGO 和 JBoss Negotiation）用于已部署应用的 SSO 所需的所有组件，但需要进行以下配置更改：



注意

显示的管理 CLI 命令假定您在运行 JBoss EAP 单机服务器。有关将管理 CLI 用于 JBoss EAP 受管域的更多详细信息，请参见 JBoss EAP *管理 CLI 指南*。

1. 配置服务器身份或主机安全域。
此安全域将容器本身身份验证到 KDC。它需要使用登录模块，该模块接受静态登录机制，因为实际用户不参与此连接。以下示例使用静态主体并引用包含凭据的 keytab 文件。

示例：创建服务器身份安全域

```
/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
```

如果使用 IBM JDK，Kerberos 模块的选项会有所不同。**jboss.security.disable.secdomain.option** 系统属性必须设置为 **true**。如需更多信息，请参阅 配置相关系统属性。此外，登录模块应配置如下：

示例：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
```

有关配置 **Kerberos** 登录模块的完整选项列表，请参阅 JBoss EAP *登录模块参考*。

2. 配置 Web 应用安全域。
Web 应用安全域用于验证单个用户到 KDC。需要至少有一个登录模块来验证用户身份。还必须有办法搜索要应用到用户的角色。这可以通过许多不同的方式来实现，例如添加 **<mapping>** 来手动将用户映射到角色，添加第二个登录模块来映射用户到角色，等等。

   以下显示了一个 Web 应用安全域示例：

   示例：创建服务器身份安全域

```
/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
```

   有关配置 **SPNEGO** 登录模块的完整选项列表，请参阅 JBoss EAP *登录模块参考*。

3. 配置相关系统属性.
JBoss EAP 提供配置与连接 Kerberos 服务器相关的系统属性的功能。根据 KDC、Kerberos 域和网络配置，以下系统属性可能也可能不是必需的：

```
<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>
```

属性	描述
java.security.krb5.kdc	KDC 的主机名。
java.security.krb5.realm	域的名称。
java.security.krb5.conf	配置 **krb5.conf** 文件的路径。

属性	描述
jboss.security.disable.secdomain.option	当设置为 **true** 时，将禁用自动添加 **jboss.security.security_domain** 登录模块选项，以登录安全域中声明的模块。使用 IBM JDK 时必须设置为 **true**。
sun.security.krb5.debug	如果为 **true**，则会启用调试模式。

注意

默认情况下，安全域中定义的每个登录模块都自动添加 **jboss.security.security_domain** 模块选项。此选项会导致登录模块出现问题，这些模块检查以确保只定义了已知的选项。IBM Kerberos 登录模块 **com.ibm.security.auth.module.Krb5LoginModule** 是其中一个模块。在启动 JBoss EAP 时，可以通过将 **jboss.security.disable.secdomain.option** 系统属性设置为 **true 来** 禁用添加此模块选项的行为。这可以通过配置 **<system-properties>**、使用管理 CLI 或管理控制台，或者将 **-Djboss.security.disable.secdomain.option=true** 添加到启动参数来实现。

有关配置系统属性的更多信息，请参阅 JBoss EAP *管理 CLI 指南*。

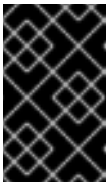
2.5. 配置 WEB 应用程序

配置完安全域后，必须将 Web 应用配置为使用这些安全域，才能启用 Kerberos 身份验证。且对应用进行了更改，便可将它部署到 JBoss EAP 实例，并开始使用 Kerberos 进行身份验证。

必须进行以下更新：

1. 配置 **web.xml** 以使用 SPNEGO 身份验证方法。
   web.xml 文件应包含以下内容：

   - 带有 **< web-resource-collection> 的 <security-constraint >**，其中包含一个 **<url-pattern>**，它映射到安全区域的 URL 模式。另外，**<security-constraint>** 也可以包含 **<auth-constraint> 来** 替代允许的角色。
   - 如果在 **<auth-constraint>** 中指定了任何角色，这些角色应在 **<security-role>** 中定义。
   - 包含 **< auth-method> 的 <login-config >**，其 *确切* 值为 **SPNEGO**。

     重要

     <auth-method> 元素需要一个以逗号分隔的特定值列表。要使 **SPNEGO** 身份验证被正确配置，*SP* **NEGO** 必须出现在 **<auth-method>** 元素中，应首先出现。在将 FORM 登录添加为 Fallback 中讨论合并其他身份验证类型。

     <security-constraint> 和 **<security-role>** 元素可让管理员根据 URL 模式和角色设置受限或不受限制的区域。这允许保护或不受保护的资源。

     示例： web.xml 文件

     ```
     <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. 配置 **jboss-web.xml** 以使用配置的安全域。
jboss-web.xml 文件应具有以下内容：

   - **<security-domain>**，用于指定用于身份验证和授权的安全域。
   - （可选） **<jacc-star-role-allow>**，可在 **web.xml** 中的 role-name 元素中使用星号字符来匹配多个角色名称。

     示例： jboss-web.xml 文件

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

3. 将 JBoss Negotiation 依赖项添加到传统 **安全子系统的部署** 中。

重要

如果您使用 **elytron** 子系统，您可以跳过这一步。

使用 SPNEGO 和 JBoss Negotiation 的 Web 应用程序需要在 **jboss-deployment-structure.xml** 中定义依赖关系，以便能够找到 JBoss Negotiation 类。由于 JBoss EAP 提供了所有必要的 JBoss 协商和相关类，应用只需将其声明为使用它们的依赖项即可。

使用 jboss-deployment-structure.xml 进行 Declare 依赖项

```
<jboss-deployment-structure>
 <deployment>
```

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

另外，也可以在 **META-INF/MANIFEST.MF** 文件中定义这个依赖项：

使用 META-INF/MANIFEST.MF 进行依赖

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

2.6. ACTIVE DIRECTORY 的其他注意事项

本节介绍如何配置 JBoss EAP 在 Microsoft Windows 服务器（Active Directory 域的一部分）上运行的 SPNEGO 身份验证所需的帐户。

在本节中，用于以 HOST_NAME 的形式访问服务器的主机名称为 ***HOST_NAME***，域称为 ***REALM***，该域称为 ***DOMAIN***，托管 JBoss EAP 实例的服务器称为 ***MACHINE_NAME***。附录.

2.6.1. 配置 Microsoft Windows 域

1. 清除现有的服务主体映射。
   在 Microsoft Windows 网络上，将自动创建一些映射。删除自动创建的映射，将服务器的身份映射到服务主体，以便正确进行协商。该映射使客户端计算机上的 Web 浏览器能够信任服务器并尝试 SPNEGO。客户端计算机通过域控制器验证以 HTTP/HOST_***NAME 形式进行的*** 映射。

   以下是删除现有映射的步骤：

   使用以下命令，列出使用计算机域注册的映射：

   ```
   setspn -L MACHINE_NAME
   ```

   使用以下命令删除现有映射：

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

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

2. 创建主机用户帐户。

   注意

   确保主机名与 ***MACHINE_NAME*** 不同。

   在本节的其余部分中，主机名指代为 ***USER_NAME***。

3. 定义 ***USER_NAME 和 HOST_NAME*** 之间的映射。
   运行以下命令来配置服务主体映射：

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

出现提示时，输入用户名的密码。

运行以下命令来验证映射：sets **pn -L *USER_NAME***.

注意

如果获得 **KrbException: 指定版本的密钥没有 JRE 中的可用** 错误，您可能需要将 Key Version 设置为 **0** : **-kvno 0**。请注意，*RE* ALM 需要全部以大写形式，而 *HOST_NAME* 应为小写。此外，*HOST_NAME* 必须是 FQDN，必须是可解析的 **A** 或 **AAAA** 记录，而不是 **CNAME** 记录。

使用 **-crypto 仅适用于** Windows Server 2008 及更高版本。对于 Windows Server 2003，您必须指定确切的设置。

4. 在安全域中定义主体。
主体可以在 **elytron** 或传统 **安全** 子系统中定义或更新至 **HTTP/HOST_*NAME*@*REALM***。

重要

如果您对任何用户进行任何修改（例如更改选项或其密码），您必须重新生成 **keytab**。

第 3 章 其他功能

3.1. 添加 FORM 登录作为 FALLBACK

JBoss EAP 及其部署的应用还可以配置 FORM 登录身份验证机制，以用作回退。这允许应用程序在不存在 Kerberos/SPNEGO 令牌的情况下显示登录页面以进行身份验证。此身份验证独立于 Kerberos 身份验证进行。因此，根据 FORM 登录回退的配置方式，用户可能需要单独的凭证来通过此方法进行身份验证。

注意

如果不存在 SPNEGO 或 NTLM 令牌，或者存在 SPNEGO 令牌，但来自其他 KDC，则可以使用回退到 FORM 登录。

3.1.1. 更新应用程序

为 FORM 登录配置应用程序作为回退，需要执行以下步骤：

1. 将 JBoss EAP 和 Web 应用配置为使用 Kerberos 和 SPNEGO.
   有关配置 JBoss EAP 和 Web 应用以使用 Kerberos 和 SPNEGO 进行身份验证和授权所需的步骤，请参阅如何使用 Kerberos 设置 SSO。

2. 添加登录和错误页面。
   要使用 FORM 登录，需要登录和错误页面。这些文件添加到 Web 应用中，并在身份验证过程中使用。

   示例： login.jsp File

```
<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>
```

   示例： Error.jsp File

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

3. 修改 **web.xml**。
向 Web 应用中添加登录和错误页面后，必须更新 **web.xml**，以将这些文件用于 FORM 登录。*确切的*值 **FORM** 必须添加到 **<auth-method>** 元素中。因为 **<auth-method>** 需要一个用逗号分开的列表和顺序，所以 **<auth-method>** *的确切*值必须更新为 **SPNEGO,FORM**。另外，**<form-login-config>** 元素必须添加到 **<login-config>** 中，以及作为 <form- **login-page> 和 <form-error-page>** 元素指定的登录路径和错误页面。

示例：更新的 web.xml 文件

```
<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. 更新 Elytron 子系统

1. 在 **http-authentication-factory** 中添加 FORM **身份验证** 机制。
您可以使用您为基于 kerberos 的身份验证配置的现有 **http-authentication-factory**，以及用于 FORM **身份验证** 的额外机制。

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

2. 添加额外的回退主体。
基于 kerberos 的身份验证的现有配置应已配置了安全域，用于将主体从 kerberos 令牌映射到应用的角色。您可以向该域添加用于回退身份验证的其他用户。例如，如果您使用的是 **filesystem-realm**，只需创建一个具有适当角色的新用户：

```
/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. 更新传统安全子系统

如果您使用 JBoss EAP 中的传统 **安全** 子系统，您必须更新安全域以进行回退身份验证。

Web 应用安全域必须配置为支持回退登录机制。这需要以下步骤：

1. 添加新的安全域，以充当回退身份验证方法。
2. 向指向回退域的 Web 应用安全域添加 **usernamePasswordDomain** 模块选项。

示例：使用回调安全域配置安全域

```
/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
```

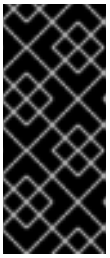
3.2. 使用 KERBEROS 保护管理接口

除了在安全域中提供 Kerberos 身份验证外，JBoss EAP 还提供使用 Kerberos 保护管理接口的功能。

3.2.1. 使用 Elytron 通过 Kerberos 保护管理接口

为 HTTP 管理接口配置 Kerberos 身份验证：

1. 按照 为应用配置 Kerberos 身份验证 的说明创建执行 Kerberos 身份验证 **的 http-authentication-factory**。



重要

使用管理接口配置 Kerberos 身份验证时，务必要密切关注为 JBoss EAP 配置的服务主体，以针对 KDC 进行身份验证。此服务主体采用 **service-name/hostname** 的形式。在针对基于 Web **的** 管理控制台进行身份验证时，JBoss EAP 要求 **HTTP 作为服务名称，如 HTTP/localhost，作为** 管理 CLI 的服务名称（如 **remote/localhost** ）。

2. 更新管理 HTTP 接口，以使用 **http-authentication-factory**。

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

为管理 CLI 配置 SASL 身份验证的 Kerberos 身份验证：

1. 按照 为应用配置 Kerberos 身份验证 以创建安全域和 **kerberos-security-factory** 的相同说明。

2. 将 **GSSAPI** 添加到 **可配置-sasl-server-factory.**

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

3. 创建一个 **sasl-authentication-factory**，它使用安全域和 **kerberos-security-factory**。

示例： 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. 更新管理 SASL 接口，以使用 **sasl-authentication-factory**。

示例：更新 sasl-authentication-factory

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

reload
```

3.2.2. 使用传统核心管理身份验证使用 Kerberos 保护管理接口

要使用旧的核心管理身份验证在管理接口中启用 Kerberos 身份验证，必须执行以下步骤：

注意

显示的管理 CLI 命令假定您在运行 JBoss EAP 单机服务器。有关将管理 CLI 用于 JBoss EAP 受管域的更多详细信息，请参见 JBoss EAP *管理 CLI 指南*。

1. 启用相关的系统属性。
如上一节中所述，启用任何所需的 JBoss EAP 系统属性，以连接 Kerberos 服务器。

2. 将 Kerberos 服务器身份添加到安全域：

在安全域中可以使用 Kerberos 身份验证之前，必须先添加与 Kerberos 服务器的连接。以下示例演示了如何将 Kerberos 服务器身份添加到现有的管理域。您需要将 **service-name**、hostname**和 MY-REALM** 替换为适当的值。

将服务器身份添加到安全域的 CLI 示例

```
/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

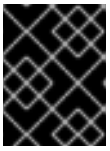
使用管理接口配置 Kerberos 身份验证时，务必要密切关注为 JBoss EAP 配置的服务主体，以针对 KDC 进行身份验证。此服务主体采用 **service-name/hostname** 的形式。在针对基于 Web **的** 管理控制台进行身份验证时，JBoss EAP 要求 **HTTP 作为服务名称，如 HTTP/localhost，作为** 管理 CLI 的服务名称（如 **remote/localhost** ）。

3. 更新安全域中的身份验证方法。
正确配置了 Kerberos 服务器身份后，需要更新安全域中的身份验证方法以使用它。

示例：在安全域中添加 Kerberos 身份验证

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

reload
```



重要

根据您在安全域中定义的身份验证机制的顺序，JBoss EAP 在访问管理接口时将尝试按该顺序对用户进行身份验证。

4. 通过 Kerberos 保护这两个接口.
如果您想同时使用 Kerberos 保护基于 Web 的管理控制台和管理 CLI，则需要为每个 Kerberos 配置 Kerberos 服务器身份。若要添加其他身份，可使用以下命令：

```
/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. 连接到管理接口

在尝试连接到管理接口之前，您需要有一个有效的 Kerberos 票据。如果安全域无法通过 Kerberos 验证用户，在使用传统安全解决方案时，它将尝试使用 **<authentication>** 元素中指定的任何方法验证用

户。**elytron** 子系统的行为与传统安全解决方案类似。如果 Kerberos 身份验证机制失败，身份验证将回退到您在保护管理界面的身份验证工厂中定义的任何其他机制。通常，DIGEST 或 BASIC 用作回退。

当您使用浏览器连接到基于 Web 的管理控制台时，安全域将尝试基于该票据进行身份验证。

连接到管理 CLI 时，您需要使用 **-Djavax.security.auth.useSubjectCredsOnly=false** 参数，因为这允许 GSSAPI 实施利用操作系统级别管理的身份。您可能需要根据环境的设置方式使用以下参数：

-Djava.security.krb5.realm=*REALM_NAME*

指定 realm 名称。

-Djava.security.krb5.kdc=*KDC_HOSTNAME*

指定 KDC 的位置。

--no-local-auth

禁用本地身份验证.如果您尝试连接运行 脚本的同一计算机上运行的 JBoss EAP 实例，这非常有用。

命令示例

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

警告

如果在客户端和服务器之间使用 HTTP 代理，则必须注意避免将不同经过身份验证的客户端共享同一服务器之间的经过身份验证的连接。如果不被允许，则服务器可以轻松丢失安全性上下文关联跟踪。正确遵循客户端到服务器身份验证完整性的代理将在代理的 HTTP 响应中为客户端 **提供Proxy-support: Session-Based-Authentication** HTTP 标头。客户端 *不得* 通过代理利用 SPNEGO HTTP 身份验证机制，除非代理提供来自服务器的 **401 Unauthorized** 响应。

3.3. 用于提升的 KERBEROS 身份验证集成

除了使用 Kerberos 保护管理界面和 Web 应用程序外，您还可以为通过远程访问的服务配置 Kerberos 身份验证，如 Jakarta Enterprise Beans。

还需要配置 Kerberos 的系统属性。如需更多信息，请参阅 配置 Elytron 子系统。

3.3.1. 使用传统安全域进行 Kerberos 身份验证集成

要配置 Kerberos 身份验证，您需要执行以下操作：

1. 使用 remoting 和 **RealmDirect**配置安全域
   您需要配置安全域，以供通过远程访问的服务使用。此安全域需要同时使用 **Remoting** login 模块和 **RealmDirect** 登录模块，如 **RealmDirect 或 RealmUsersRoles**。本质上，它应当与默认提供 **的其他** 安全域非常相似。有关每个登录模块的具体配置选项的详情，请查看 JBoss EAP *登录模块参考*。

   示例：带有远程和 RealmDirect 登录模块的安全域

   ```
   /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. 为 Kerberos 身份验证配置安全域.
   有关使用 Kerberos 身份验证设置安全域 的信息，请参见"通过 Kerberos 保护管理接口 "一节。

   示例： Security Realm

```
/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. 在远程子系统 **中配置 HTTP** 连接器。
   此外，您需要在远程子系统 **中配置** HTTP 连接器，以使用新创建的安全域。

   示例：删除子系统

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

4. 配置 服务的安全性。
   您还必须设置使用远程接口访问的服务以 **进行保护**。这将因服务而异。例如，对于 Jakarta Enterprise Beans，您可以使用 **@SecurityDomain** 和 **@RolesAllowed** 注释。

3.3.2. 使用 Elytron 进行 Kerberos 身份验证集成

可以为 Kerberos 或 GSSAPI SASL 身份验证定义 Elytron 安全域来远程进行身份验证。

1. 定义要从中加载身份的安全域。它用于分配角色。

```
/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. 为服务器身份定义 Kerberos 安全工厂。

```
/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. 定义安全域，以及 SASL 身份验证工厂。

```
/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. **在** 远程子系统中，使用创建的 **sasl-authentication-factory** 将它启用以进行远程处理。

   CLI 命令示例

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

5. 配置 服务的安全性。
   如果您引用了 Jakarta Enterprise Beans 中的安全域，您必须指定映射到 Elytron 安全域的 **application-security-domain**。例如，通过 Jakarta Enterprise Beans，您可以使用 **@SecurityDomain** 注释。

   CLI 命令示例

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

将 Jakarta 身份验证对象用于身份关联不再被支持。希望以编程方式管理 Jakarta Enterprise Beans 调用的 Kerberos 身份的客户端应迁移并直接使用 **AuthenticationConfiguration** API，如下所示：

```
// 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 Jakarta Enterprise Bean and invokes a method on it
Callable<Void> callable = () -> {
...
};
```

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

创建 **AuthenticationConfiguration** 时会调用 **使用GSSCredential（getGSSCredential（））**。已有权访问 Jakarta Authentication Subject 的客户端代码可以轻松转换以获取 **GSSCredential**，如下所示：

```
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;
        }
    });
}
```

更新于 2024-02-09