Red Hat build of OpenJDK 11

Red Hat build of OpenJDK 11.0.17 发行注记

Last Updated: 2024-05-10

法律通告

摘要

红帽构建的 OpenJDK 11.0.17 发行注记 文档概述了红帽构建的 OpenJDK 11 中的新功能以及潜在的已知问题列表和可能的临时解决方案。

目录

前言

Open Java Development Kit (OpenJDK)是 Java Platform, Standard Edition (Java SE)的一个免费的开源实现。红帽构建的 OpenJDK 在三个版本中提供：8u、11u 和 17u。

红帽构建的 OpenJDK 软件包在 Red Hat Enterprise Linux 和 Microsoft Windows 上提供，并作为红帽生态系统目录中的 JDK 和 JRE 提供。

提供有关红帽构建的 OPENJDK 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. 在 **Summary** 中输入问题的简短描述。
3. 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。
4. 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

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

第 1 章 红帽构建的 OPENJDK 支持政策

红帽在其产品中支持选择版本的 OpenJDK 的主版本。为实现一致性，这些版本与 Oracle JDK 的长期支持(LTS)指定的版本相同。

自首次引入该版本起，红帽构建的 OpenJDK 主版本将最少提供六年的支持。如需更多信息，请参阅 OpenJDK 生命周期和支持政策

注意

RHEL 6 于 2020 年 11 月结束其生命周期。因此，红帽构建的 OpenJDK 不支持 RHEL 6 作为支持的配置。

第 2 章 与上游 OPENJDK 11 的不同

Red Hat 在 Red Hat Enterprise Linux (RHEL)中构建 OpenJDK 包含了来自 OpenJDK 上游发行版的许多结构更改。红帽构建的 Microsoft Windows 版本尝试尽快遵循 RHEL 更新。

以下列表详细介绍了 OpenJDK 11 最显著的红帽构建变化：

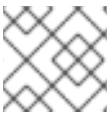
- FIPS 支持。Red Hat build of OpenJDK 11 会自动检测 RHEL 是否处于 FIPS 模式，并自动配置红帽构建的 OpenJDK 11 以在该模式下运行。此更改不适用于适用于 Microsoft Windows 的红帽构建的 OpenJDK 构建。
- 加密策略支持。红帽构建的 OpenJDK 11 从 RHEL 获取启用的加密算法和密钥大小限制列表。这些配置组件由传输层安全(TLS)加密协议、证书路径验证和任何签名的 JAR 使用。您可以设置不同的安全配置集来平衡安全性和兼容性。此更改不适用于适用于 Microsoft Windows 的红帽构建的 OpenJDK 构建。
- RHEL 上的红帽构建的 OpenJDK 会动态链接到原生库，如 **zlib** 用于归档格式支持，**libjpeg-turbo**、**libpng** 和 **giflib** 用于镜像支持。RHEL 还动态链接 **Harfbuzz** 和 **Freetype** 用于字体渲染和管理。
- **src.zip** 文件包含红帽构建的 OpenJDK 附带的所有 JAR 库的源。
- RHEL 上的红帽 OpenJDK 构建使用系统范围的时区数据文件作为时区信息的来源。
- RHEL 上的红帽构建的 OpenJDK 使用系统范围的 CA 证书。
- Microsoft Windows 上的红帽构建的 OpenJDK 包括 RHEL 的最新可用时区数据。
- Microsoft Windows 上的红帽构建的 OpenJDK 使用 RHEL 的最新可用 CA 证书。

其他资源

- 有关检测系统是否处于 FIPS 模式的更多信息，请参阅 Red Hat RHEL 计划 JIRA 中的增强系统 FIPS 检测示例。
- 有关加密策略的更多信息，请参阅使用系统范围的加密策略。

第 3 章 RED HAT BUILD OF OPENJDK 功能

最新的 Red Hat build of OpenJDK 11 发行版本可能包括新功能。另外，最新版本可能会增强、弃用或删除来自以前红帽构建的 OpenJDK 11 版本的功能。



注意

有关所有其他更改和安全修复，请参阅 OpenJDK 11.0.17 发行版本。

红帽构建的 OpenJDK 新功能及改进

查看以下发行注记以了解红帽构建的 OpenJDK 11.0.17 发行版本中包含的新功能和功能增强：

禁用 cpu.shares 参数

在红帽构建的 OpenJDK 11.0.17 版本之前，红帽构建的 OpenJDK 使用了 **cpu.shares** 参数的不正确的解释，它属于 Linux 控制组，也称为 **cgroups**。参数可能会导致 Java 虚拟机(JVM)使用比可用的 CPU 少，这可能会在容器内运行时影响 JVM 的 CPU 资源和性能。

红帽构建的 OpenJDK 11.0.17 发行版本将 JVM 配置为在决定线程池的线程数量时不再使用 **cpu.shares** 参数。如果要恢复此配置，请在 JVM 启动时传递 **-XX:+UseContainerCpuShares** 参数。

注意

-XX:+UseContainerCpuShares 参数是一个已弃用的功能，可能会在以后的红帽构建的 OpenJDK 版本中删除。

请参阅 JDK-8281181 (JDK Bug System)。

jdk.httpserver.maxConnections system property

红帽构建的 OpenJDK 11.0.17 添加了一个新的系统属性 **jdk.httpserver.maxConnections**，它会修复为 **HttpServer** 服务指定任何连接限制的安全问题，这可能会导致接受的连接和建立的连接无限期保持打开。

您可以使用 **jdk.httpserver.maxConnections** 系统属性更改 **HttpServer** 服务，其行为如下：

- 设置 **0** 或负值（如 **-1** ）来为该服务指定连接限制。
- 设置一个正值，如 **1**，使服务能够根据当前已建立的连接数检查任何接受的连接。如果达到该服务的建立连接，服务会立即关闭接受的连接。

请参阅 JDK-8286918 (JDK Bug System)。

使用 JFR 监控对象的反序列化

现在，您可以使用 JDK Flight Recorder (JFR)监控对象的反序列化。默认情况下，红帽构建的 OpenJDK 11.0.17 禁用 JFR 的 **jdk.deserialization** 事件设置。您可以通过更新 JFR 配置中的 **event-name** 元素来启用此功能。例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration version="2.0" description="test">
 <event name="jdk.Deserialization">
 <setting name="enabled">true</setting>
 <setting name="stackTrace">false</setting>
 </event>
</configuration>
```

启用 JFR 并配置 JFR 以监控反序列化事件后，JFR 会在受监控的应用程序尝试反序列化对象时创建一个事件。然后，JFR 的序列化过滤器机制可以确定是否接受或拒绝被监控应用程序中的反序列化对象。

请参阅 JDK-8261160 (JDK Bug System)。

SHA-1 Signed JARs

随着红帽构建的 OpenJDK 11.0.17 发行版本，默认使用 **SHA-1** 算法签名的 JAR，并像未签名一样对待。这些限制适用于以下算法：

- 用于摘要、签名和可选时间戳的算法。
- 代码签名者和 Timestamp Authority 的证书链中的证书的签名和摘要算法，以及任何证书撤销列表(CRL)或在线证书状态协议(OCSP)响应，用于验证这些证书是否已撤销。

另外，限制适用于签名的 Java Cryptography 扩展(JCE)供应商。

要降低之前时间戳的 JAR 的兼容性风险，限制不适用于 **2019 年 1 月 01** 日前使用 **SHA-1** 算法签名的任何 JAR 和时间戳。在以后的 Red Hat build of OpenJDK 发行版本中可能会删除这个例外。

要确定您的 JAR 文件是否受到限制的影响，您可以在 CLI 中发出以下命令：

```
$ jarsigner -verify -verbose -certs
```

在上一命令的输出中，搜索 **SHA1** 实例、**SHA-1** 或 **禁用**。此外，搜索指示 JAR 将被视为未签名的警告消息。例如：

```
Signed by "CN="Signer""
Digest algorithm: SHA-1 (disabled)
Signature algorithm: SHA1withRSA (disabled), 2048-bit key

WARNING: The jar will be treated as unsigned, because it is signed with a weak algorithm that is now disabled by the security property:

jdk.jar.disabledAlgorithms=MD2, MD5, RSA keySize < 1024, DSA keySize < 1024, SHA1 denyAfter 2019-01-01
```

考虑使用更强大的算法替换或重新签名受新限制影响的任何 JAR。

如果您的 JAR 文件受这个限制的影响，您可以删除算法并使用更强大的算法（如 **SHA-256** ）重新签署该文件。如果要删除红帽构建的 OpenJDK 11.0.17 的 **SHA-1** 签名 JAR 限制，并接受安全风险，您可以完成以下操作：

1. 修改 **java.security** 配置文件。或者，您可以保留此文件，并使用所需配置创建另一个文件。
2. 从 **jdk.certpath.disabledAlgorithms** 安全属性中删除 **SHA1 使用 SignedJAR & denyAfter 2019 01 011** 条目。
3. 从 **jdk.jar.disabledAlgorithms** 安全属性中删除 **SHA1 denyAfter 2019-01-01** 条目。

注意

java.security 文件中的 **jdk.certpath.disabledAlgorithms** 的值可能会被 RHEL 8 和 9 上的系统安全策略覆盖。系统安全策略使用的值可在文件 **/etc/crypto-policies/back-ends/java.config** 中看到，并通过在 java.security 文件中将 **security.useSystemPropertiesFile** 设置为 false 来禁用，或者将 **-Djava.security.disableSystemPropertiesFile=true** 设置为 JVM。这些值不会在此发行版本中修改，因此对于以前的红帽构建的 OpenJDK 版本，这些值保持不变。

有关配置 **java.security** 文件的示例，请参阅 为 OpenShift （红帽客户门户网站）覆盖 JBoss EAP 的 **java.security** 属性。

请参阅 JDK-8269039 (JDK Bug System)。

控制 HTTPURLConnection的 keep-alive 行为的系统属性

红帽构建的 OpenJDK 11.0.17 发行版本包括以下新系统属性，可用于控制 **HTTPURLConnection** 的 **keep-alive** 行为：

- **http.keepAlive.time.server**，用于控制与服务器的连接。
- **http.keepAlive.time.proxy**，用于控制到代理的连接。

在红帽构建的 OpenJDK 11.0.17 发行版本之前，带有未指定 **keep-alive** 时间的服务器或代理可能会导致闲置连接在由硬编码的默认值定义的期间保持打开。

使用红帽构建的 OpenJDK 11.0.17，您可以使用系统属性更改 **keep-alive** 时间的默认值。**keep-alive** 属性通过更改服务器或代理的 HTTP **keep-alive** 时间来控制此行为，以便红帽构建的 OpenJDK 的 HTTP 协议处理器在指定秒数后关闭闲置连接。

在红帽构建的 OpenJDK 11.0.17 版本前，以下用例会导致 **HTTPURLConnection** 的特定 **keep-alive** 行为：

- 如果服务器指定了 **Connection:keep-alive** 标头，服务器的响应包含 **Keep-alive:timeout=N**，则客户端上的红帽构建的 OpenJDK **keep-alive** 缓存使用 **N** 秒超时，其中 **N** 是整数值。
- 如果服务器指定了 **Connection:keep-alive** 标头，但服务器的响应不包含 **Keep-alive:timeout=N** 的条目，则客户端上的红帽构建的 OpenJDK **keep-alive** 缓存使用 **60** 秒用于代理，**5** 秒用于服务器。
- 如果服务器没有指定 **Connection:keep-alive** 标头，则客户端上的 OpenJDK **keep-alive** 缓存的红帽构建会对所有连接使用 5 秒的超时时间。

红帽构建的 OpenJDK 11.0.17 发行版本维护之前描述的行为，但您现在可以使用 http.keepAlive.time.**server 和 http.keepAlive.time.proxy** 属性来指定第二个和第三个用例中的超时，而不必依赖默认设置。

注意

如果您设置了 **keep-alive** 属性，且服务器为 **Keep-Alive** 响应标头指定一个 **keep-alive** 时间，则 HTTP 协议处理程序将使用服务器指定的时间。对于代理，这个情况是相同的。

请参阅 JDK-8278067 (JDK Bug System)。

更新了默认的 PKCS libpmem MAC 算法

Red Hat build of OpenJDK 11.0.17 更新 PKCS TOTP 密钥存储的默认消息身份验证代码(MAC)算法，以使用 **SHA-256** 加密哈希功能而不是 **SHA-1** 功能。**SHA-256** 功能为保护数据提供了更强大的方法。

您可以在 **keystore.pkcs12.macAlgorithm** 和 **keystore.pkcs12.macIterationCount** 系统属性中查看此更新。

如果您使用这个更新的 MAC 算法创建密钥存储，且您试图将密钥存储与红帽构建的 OpenJDK 版本一起使用，而不是红帽构建的 OpenJDK 11.0.12，您将收到 **java.security.NoSuchAlgorithmException** 信息。

要将之前的密钥存储用于红帽构建的 OpenJDK 版本，该版本早于 OpenJDK 11.0.12，将 **keystore.pkcs12.legacy** 系统属性设置为 **true** 以恢复 MAC 算法。

请参阅 JDK-8267880 (JDK Bug System)。

弃用和删除的功能

查看以下发行注记以了解在 Red Hat build of OpenJDK 11.0.17 发行版本中已弃用或删除的预先存在的功能：

弃用的 Kerberos 加密类型

红帽构建的 OpenJDK 11.0.17 弃用了 **3-hmac-sha1** 和 **rc4-hmac** Kerberos 加密类型。默认情况下，红帽构建的 OpenJDK 11.0.17 禁用这些加密类型，但您可以通过完成以下操作来启用它们：

- 在 **krb5.conf** 配置文件中，将 **allow_weak_crypto** 选项卡设置为 **true**。此配置还启用其他加密类型，如 **des-cbc-crc** 和 **des-cbc-md5**。

警告

在应用此配置前，请考虑启用所有弱 Kerberos 加密类型的风险，比如在您的 Kerberos 身份验证机制中引入弱加密算法。

您可以通过在以下 **krb5.conf** 配置文件中明确列出加密类型来禁用弱加密类型的子集：

- **default_tkt_enctypes**
- **default_tgs_enctypes**
- **permitted_enctypes**

请参阅 JDK-8139348 (JDK Bug System)。

第 4 章 与本发行版本相关的公告

以下公告针对程序错误修正和 CVE 修复进行了程序错误修正和 CVE 修复：

- RHSA-2022:7008
- RHSA-2022:7009
- RHSA-2022:7010
- RHSA-2022:7011
- RHSA-2022:7012
- RHSA-2022:7013
- RHSA-2022:7052
- RHSA-2022:7054

更新于 2024-05-10