Apache Tomcat 安全基线检查

1. 既无权访问非必须的文件，又不能执行非必须的程序

威胁等级：High

规则描述：Java 运行时根据运行它的用户授予安全权限。当 Tomcat 以系统管理员身份或作为系统服务运行时，Java 运行时取得了系统用户或系统管理员所具有的全部权限。这样一来，Java 运行时就取得了所有文件夹中所有文件的全部权限。并且 Servlets(JSP 在运行过程中要转换成 Servlets)取得了同样的权限。所以 Java 代码可以调用 Java SDK 中的文件 API 列出文件夹中的全部文件，删除任何文件，最大的危险在于以系统权限运行一个程序。

审计描述：执行命令：ps -ef | grep -v grep | grep tomcat，查看第一项 user 是不是 root。

检测描述：检查运行 tomcat 的用户是否合法

期望结果：非 root

检测结果：如下图所示

修改建议：当安装账户为超级管理员时，创建低权限的账号运行 Tomcat，操作步骤如下

--新增 tomcat 用户

useradd tomcat

--将 tomcat 目录 owner 改为 tomcat

chown -R tomcat: /usr/local/apache-tomcat-8.5.82

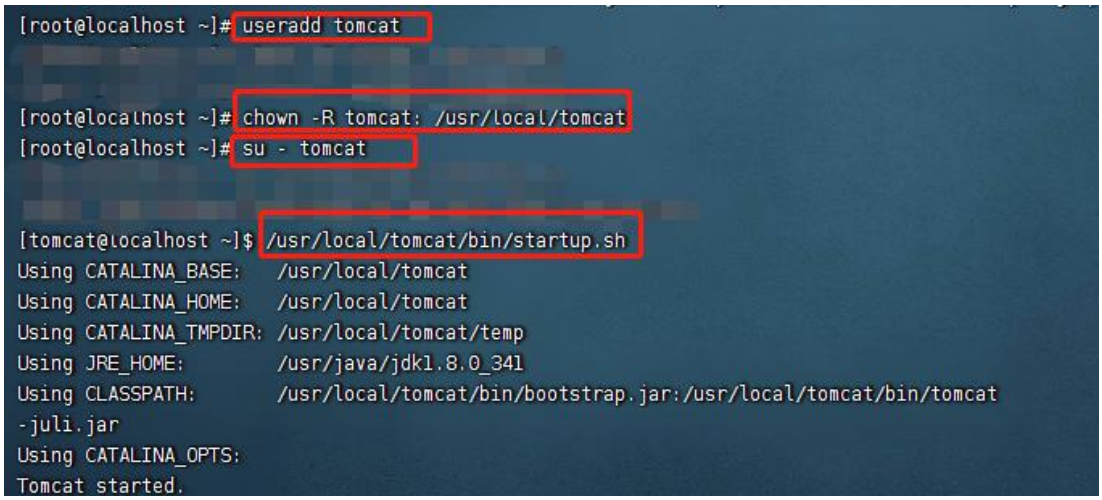
--停止原来的 tomcat 服务

--切换到 tomcat 用户

su - tomcat

--重新启动 tomcat

/usr/local/tomcat/bin/startup.sh



2. 删除安装过程文件、缺省安装的管理帮助文件及用户测试类的非必需文件

威胁等级：Medium

规则描述：默认的示例或测试应用容易被远程访问或执行，给系统带来相当的危害。

审计描述：

检查以下目录是否存在非必需的管理应用和帮助应用：tomcat\webapps*

Tomcat6 之前的版本的版本还需要检查以下目录：tomcat\server\webapps*，

检查系统示例程序和网页：tomcat\webapps\examples

检测描述：检查不存在安装过程文件、管理帮助文件及用户测试类文件。

期望结果：不存在下列文件：webapps/js-examples, webapps/servlet-example, webapps/webdav, webapps/docs, webapps/balancer, webapps/ROOT/admin, webapps/examples, /webapps/host-manager, /webapps/manager, conf/Catalina/localhost/host-manager.xml, conf/Catalina/localhost/manager.xml

检测结果：

修改建议：删除 Tomcat 示例程序和目录、管理控制台等，即从 Tomcat 根目

录的 webapps 目录，移出或删除 docs、examples、host-manager、manager 目录。

```
[root@localhost webapps]# rm -rf /usr/local/tomcat/webapps/docs
[root@localhost webapps]# rm -rf /usr/local/tomcat/webapps/examples
[root@localhost webapps]# rm -rf /usr/local/tomcat/webapps/host-manager
[root@localhost webapps]# rm -rf /usr/local/tomcat/webapps/manager
[root@localhost webapps]# ls
ROOT
```

3. 禁用应用程序自动部署功能

威胁等级：High

规则描述：默认情况下，tomcat 启动时，会自动部署$appBase 下面的所有应用。例如，当$appBase 的值为 webapps 时，那么任意一个应用，只要被放进 webapps 目录下，在 tomcat 启动时，都会被发布。这有可能导致恶意或者未经测试的应用程序被自动部署在服务器上。因此，这里必须禁用掉 Tomcat 的自动部署功能。

审计描述：检查配置文件$tomcat/conf/server.xml，Host 是否将 autoDeploy 属性设置为 false。

检测描述：Host 节点属性 autoDeploy 设置为 false

期望结果：autoDeploy=false

检测结果：

```
<Host name="localhost"  appBase="webapps"
      unpackWARs="true" autoDeploy="true">
```

修改建议：

修改配置文件$tomcat/conf/server.xml 如下：

autoDeploy="false" //自动部署

4. 禁用不必要的 http 方法,DELETE、PUT、TRACE、HEAD、OPTIONS 等

威胁等级：High

规则描述：Tomcat 服务器提供默认 http 方法包括 GET、HEAD、POST、PUT、DELETE、OPTIONS。在这些方法中，PUT、DELETE 方法很少被使用到，并且极易

被利用来进行攻击。

审计描述：

1.在$tomcat/conf/web.xml 检查 readonly 参数的值是否为 true，缺省值为 true；

2.在$tomcat/conf/server.xml 检查 allowTrace 的值是否为 false，缺省值为 false；

3. 检查 $tomcat/conf/web.xml 文件的<web-app>节点中是否配置了<web-resource-collection>节点，<http-method>节点是否为 OPTIONS HEAD PUT DELETE TRACE。

检测描述：web-resource-collection 节点必须包含的方法

期望结果：OPTIONS HEAD PUT DELETE TRACE

检测结果：

检查 web 文档之后，无 readonly 参数，无<web-resource-collection>节点，无<http-method>节点。

修改建议：

1.禁用 DELETE 和 PUT 的方法：

在$tomcat/conf/web.xml 检查 readonly 参数的值是否为 true:

```
<init-param>
    <param-name>readonly</param-name>
    <param-value>true</param-value>
</init-param>
```

备注，如果**不存在 readonly 参数**，则不用配置，因为该参数的默认值为 true；如果配置了该参数，则需要确保参数值为 true。

2.禁用 Trace 的方法：在$tomcat/conf/server.xml 禁用 trace 方法，即配置 allowTrace 为 false：

```
<Connector port="80" maxThreads="150" connectionTimeout="20000" redirectPort="8443" allowTrace="false"/>
```

备注：如果**不存在 allowTrace 参数**，则不用配置，因为该参数的默认值为 false；如果配置了该参数，需要确保参数值为 false。

3.在$tomcat/conf/web.xml 文件的<web-app>节中增加以下内容：

```
<security-constraint>
    <web-resource-collection>
        <url-pattern>/*</url-pattern>
        <http-method>OPTIONS</http-method>
        <http-method>HEAD</http-method>
        <http-method>PUT</http-method>
        <http-method>DELETE</http-method>
        <http-method>TRACE</http-method>
    </web-resource-collection>
    <auth-constraint>
    </auth-constraint>
</security-constraint>
```

说明：这里的<url-pattern>/*</url-pattern>和策略可以根据实际业务来进行配置。

备注：web-resource-collection 节点必须包含的方法：OPTIONS HEAD PUT DELETE TRACE。如果确实是业务需要使用，则不强制要求。

5. 定制 Tomcat 出错信息

威胁等级：Medium

规则描述：Tomcat 发生服务器端出错时（例如：Tomcat 在找不到网页时，会报 404 错误），错误页面上会附带当前服务器版本号，还可能泄露其他服务器端信息。黑客可以通过版本号，查询当前 Tomcat 服务器的默认配置信息，以及该版本的安全漏洞。

审计描述：

检查配置文件$tomcat/conf/web.xml 的倒数第二行（即</web-app>之前的那一行）是否有如下内容：

```
<error-page>
 <error-code>400</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
```

```
 <error-code>401</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>402</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>403</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>404</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>405</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>406</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>407</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>413</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>414</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>500</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
 <error-code>501</error-code>
 <location>/error.htm</location>
</error-page>
<error-page>
```

<exception-type>java.lang.Throwable</exception-type>
<location>/error.htm</location>
</error-page>

检测描述：检查 Tomcat 是否定制出错信息
期望结果：

```
<web-app>
    <error-page>
        <location>*</location>
    </error-page>...
</web-app>
```

检测结果：未配置<error-page>
修改建议：

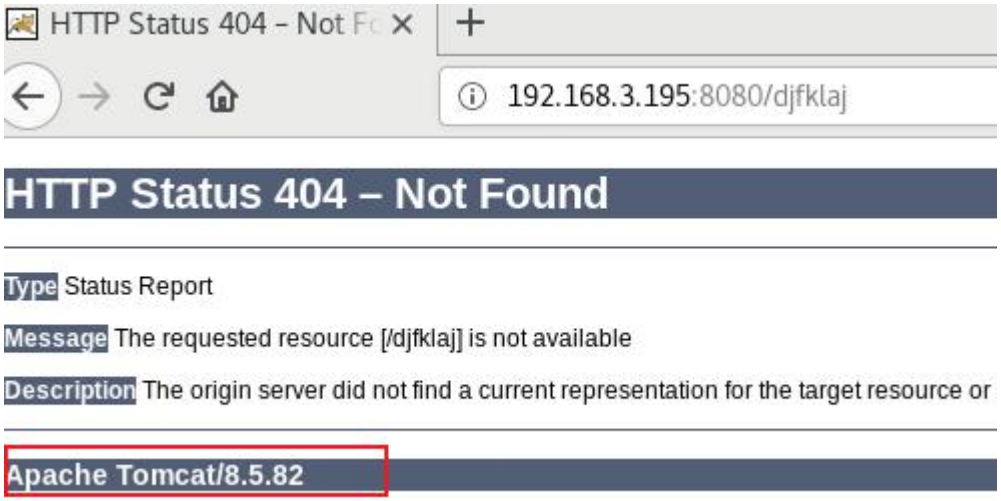
在配置文件$tomcat/conf/web.xml 的倒数第二行（即</web-app>之前的那一行）添加如上审计内容。

备注：

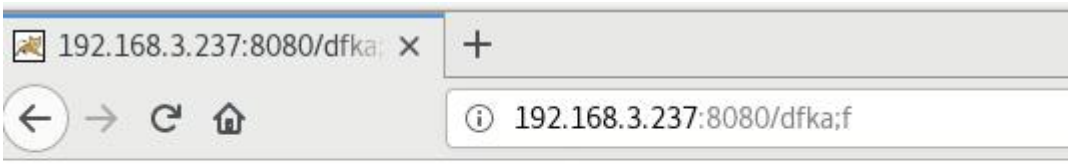
error.html 文件要拷贝到各 web 应用的根目录，可以在 error.html 文件中自定义出错提示信息，error.html 的大小必须大于 512 个字节，否则 IE 会自动调用自己的友好提示界面。特别提醒，禁止给响应码 302 配置错误页面，否则所有重定向都将指向该错误页面。<location>的配置值是自定义错误页面相对于当前 Web 应用的根目录的路径，需要根据自定义错误页面的实际路径配置。

使用 Eclipse 集成 Tomcat 后，如果指定了错误页面，会看不到异常抛出，无法定位错误。可以在开发阶段先注释掉此段，在发布时再取消注释。

未设置，暴露版本信息。



设置后效果



6. 更改 Tomcat 服务器默认 shutdown 端口号和命令

威胁等级：High

规则描述：Tomcat 服务器提供默认 shutdown 端口(8005)和命令(SHUTDOWN)，很容易被黑客捕获利用关闭服务器，进而威胁到服务器和应用等。

审计描述：检查$tomcat/conf/server.xml 文件的 port 值是否为 8005，shutdown 的值是否为大于 12 位的任意字符串。如果将 port 值改成-1 表示不使用 shutdown 端口，结果为通过。

检测描述：检查 shutdown 命令是否满足复杂度要求

期望结果：shutdown 命令满足复杂度要求

检测结果：

```
<Server port="8005" shutdown="SHUTDOWN">
```

修改建议：

1．使用 shutdown.sh 关闭 tomcat。

2．修改$tomcat/conf/server.xml 文件，更改默认端口和 shutdown 命令为其他大于 8 位的任意字符串，例如：

<Server port="2324" shutdown="Rea!!yC0mplexW0rd">//这里 shutdown 命令建议 12 个字符，包含大小写字母、数字和特殊字符。

3．使用 startup.sh 启动 tomcat。

备注：默认的 8005 端口为 Tomcat 的 shutdown 端口，在 linux 下如果改成-1 表示不使用 shutdown 端口。

7. 限制 http 请求的消息主体和消息头的大小

威胁等级：Medium

规则描述：此指令给了服务器管理员更大的可控性，以控制客户端不正常的请求行为。

审计描述：

检查配置文件$tomcat/conf/server.xml 中是否有如下内容：

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000" maxPostSize="10240" maxHttpHeaderSize="8192"/>
```

检测描述：检查是否限制 http 请求的消息主体和消息头的大小

期望结果：

```
<Server>
...
   <Connector ...  protocol="HTTP/1.1"   maxPostSize="10240" ... maxHttpHeaderSize="8192"... />
...
</Server>
```

检测结果：

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
<!-- A "Connector" using the shared thread pool-->
```

修改建议：

在配置文件$tomcat/conf/server.xml 中的每个 Connector 的“maxPostSize”属性为 10240，“maxHttpHeaderSize”属性为 8192：

```
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000" maxPostSize="10240" maxHttpHeaderSize="8192"/>
```

备注：推荐请求体大小限制为 10240bytes，消息头的大小限制为 8192bytes，如果产品有特殊需求，可以进行相应调整。

8. 禁止以特权方式运行应用

威胁等级：High

规则描述：以特权方式运行应用会允许应用加载管理库。

审计描述：检查所有 context.xml 文件中 Context 节点是否有 privileged 属性，如果有则值必须为 false。

检测描述：所有 context.xml 文件中 Context 节点的 privileged 属性值为 false 或者不存在该属性

期望结果：

<Context ... privileged= "false" /> or <Context ...(无 privileged 属性) />

检测结果：无 privileged 属性

修改建议：如果有 privileged 属性，在所有的 context.xml 中设置

<Context ... privileged= "false"/>

备注：该参数默认值为 false，如果未发现该参数，则不用设置。

9. 删除 Tomcat 默认管理控制台

威胁等级：Medium

规则描述：默认情况下，Tomcat 存在管理控制台，其地址一般为 http://[IP]:[Port]/admin。其用户密码，在$tomcat/conf/tomcat-users.xml 中定义。在$tomcat/webapps 下的 admin.xml 和 manager.xml 文件，定义了可以通过访问/admin 和/manager 进入控制台的通道。默认情况下，可以轻易的登录 tomcat 管理台，造成严重安全问题。

本方法适用于完全不需要使用默认控制台及相关功能的应用场景。

审计描述：

1、检查$tomcat/webapps 下不应该存在 admin.xml 和 manager.xml 文件；

2、检查$tomcat/conf/tomcat-users.xml 中不应该存在用户和密码；

3、检查$tomcat/webapps 下不应该存在 admin 和 manager 两个应用。

检测描述：检查是否删除$tomcat/webapps 下的 admin 和 manager 两个应用；

期望结果：$tomcat/webapps 下不存在 admin 和 manager

检测结果：$tomcat/webapps 下不存在 admin 和 manager

10. 禁用 webdav

威胁等级：High

规则描述：WebDAV（Web-based Distributed Authoring and Versioning）是基于 HTTP 1.1 的一个通信协议。它为 HTTP 1.1 添加了一些扩展（就是在 GET、POST、HEAD 等几个 HTTP 标准方法以外添加了一些新的方法），使得应用程序可以直接将文件写到 Web Server 上，并且在写文件时候可以对文件加锁，写完后对文件解锁，还可以支持对文件所做的版本控制。这存在一定的安全问题。Tomcat 本身是支持 WebDAV 的，虽然需要进行配置才可以启用。

审计描述：检查文件$tomcat/conf/web.xml 的 servlet 节点是否存在 webdav 值

检测描述：检查文件$tomcat/conf/web.xml 的 servlet 节点是否存在 webdav 值

期望结果：不存在

检测结果：不存在

修改建议：如果存在，则在配置文件$tomcat/conf/web.xml 中，确保下面的配置节点不存在或者处于注释状态：

```
<servlet>
    <servlet-name>webdav</servlet-name>
    <servlet-class>org.apache.catalina.servlets.WebdavServlet</servlet-class>
    ...
```

11. 禁用 Symbolic links

威胁等级：High

规则描述：Symbolic links 是用来解决不同的 web 应用程序之间共享文件的一种方式。这会造成应用之间的相互影响，一个应用的安全漏洞，有可能影响到所有关联的应用，因此，这种方式在安全方面存在极大隐患。

审计描述：

1.检查配置文件$tomcat/conf/server.xml 的每个 context 节点是否存在 a

llowLinking 属性。

2.检查配置文件$tomcat/conf/context.xml 的每个 context 节点是否存在 allowLinking 属性。

备注：在每个 web 应用的 META-INF 下检查是否存在 context.xml 文件，如果存在，请检查并确保每个 Context 节点都没有 allowLinking 属性。

检测内容：检查是否禁用 Symbolic links

期望结果：已禁用

检测结果：已禁用

修改建议：如果未禁用，1. 在配置文件$tomcat/conf/server.xml 中，确保每个 Context 节点都没有 allowLinking 属性。

2. 在配置文件$tomcat/conf/context.xml 中，确保每个 context 节点都没有 allowLinking 属性。

3. 在每个 web 应用的 META-INF 下检查是否存在 context.xml 文件，如果存在，请检查并确保每个 Context 节点都没有 allowLinking 属性。

12. 禁止配置 Tomcat 的网络连接超时时间为 0 或-1

威胁等级：High

规则描述：connectionTimeout 为网络连接超时时间毫秒数，当配置为 0 或-1 时，表示永不超时，在受到 DOS 攻击时，很快就会导致最大连接数被完全占用，进而导致 Tomcat 服务器无法访问。因此这里禁止配置 connectionTimeout 为 0 和-1，通常推荐的超时时间为 20s 和 30s。特殊情况下，请根据具体性能需求进行调优。

审计描述：检查配置文件$tomcat/conf/server.xml 中的每个 Connector 的 connectionTimeout 属性是否被置为 0 或者-1。

检测描述：查看每个 Connector 的 connectionTimeout 属性是否被置为 0 或者-1。

期望结果：未被置为 0 或-1

检测结果：未被置为 0 或-1

```
<!--
<Connector executor="tomcatThreadPool"
           port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
-->
```

13. 禁用 SSI 和 CGI 功能

威胁等级：High

规则描述：SSI 指令可以用于执行 Tomcat JVM 外部的程序，CGI 脚本可以用于执行 Tomcat 的 java 虚拟机外部的程序，所以这是极度危险的。

审计描述：确认配置文件$tomcat/conf/web.xml 中，<servlet>标签下的 SSI 和 CGI 配置均处于 xml 注释状态。

检测描述：是否禁用 SSI 和 CGI 功能

期望结果：已禁用

检测结果：已禁用

修改建议：Tomcat 默认不支持 SSI 和 CGI，修改配置文件$tomcat/conf/web.xml，<servlet>标签下的 SSI 和 CGI 配置均处于 xml 注释状态。

14. 启用 context.xml 文件的 useHttpOnly 参数，防止会话 cookie 被客户端脚本访问

威胁等级：High

规则描述：设置 cookie 为 HttpOnly，可以阻止客户端脚本访问（包括读取和修改）cookie，当支持 HttpOnly 的客户端浏览器（当前主流的浏览器都支持）检测到 Cookie 包括了 HttpOnly 标志时，浏览器返回空字符串给企图读取该 cookie 的脚本，这样 cookie 中的任何信息暴露给黑客或者恶意网站的几率将会大大降低。早期需要通过服务端代码设置 cookie 的 HttpOnly 属性值为 true，Tomcat 在 Tomcat 6.0 以上版本提供新的功能：只需要配置 context.xml 文件的 useHttpOnly 参数，即可实现对会话 cookie 设置 HttpOnly 属性，从而保护会话 cookie 不被客户端脚本访问。

审计描述：检查$tomcat/conf/context.xml 文件中，确认 useHttpOnly 属性为非 false。

期望结果：useHttpOnly 属性为 true

检测结果：默认设置 useHttpOnly 参数值为 true。

修改建议：在$tomcat/conf/context.xml 中设置 context 的参数 useHttpOnly 的值为 true，例如：

<Context useHttpOnly="true">

备注:useHttpOnly 参数适用于 Tomcat 6.0.x 的版本;Tomcat7 不需要设置，因为 Tomcat7 默认设置 useHttpOnly 参数值为 true。

```
<Context useHttpOnly="true">
</Context>
```

15. 禁用额外的路径分隔符

威胁等级：High

规则描述：允许额外路径分隔符会导致黑客访问原本不可见的应用或区域。

审计描述：

检查$tomcat/bin/catalina.sh 文件中是否存在如下参数：

-Dorg.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH=false

-Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=false

如果存在，则值必须为 false。

期望结果：

-Dorg.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH=false

-Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=false

检测结果：启动脚本中未发现这两个参数

修改建议：

在启动脚本$tomcat/bin/catalina.sh 的配置行 JAVA_OPTS="$JAVA_OPTS ..."中添加如下命令：

-Dorg.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH=false

-Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=false，

即 JAVA_OPTS="$JAVA_OPTS -Dorg.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH=false -Dorg.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH=false"，重启服务

备注：这两个参数默认值为 false。**如果在启动脚本中未发现这两个参数，则不用添加。**

16. 禁用自定义头状态信息

威胁等级：High

规则描述：允许将用户提供的数据放入头中可能导致 XSS 攻击。

审计描述：

检查$tomcat/bin/catalina.sh 文件中是否存在 -Dorg.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER 字段，如果有则值必须为 false

检测描述：-Dorg.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER 是否为 false

期望结果：Dorg.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER=false

检测结果：不存在该字段

修改建议：

在启动脚本$tomcat/bin/catalina.sh 的配置行 JAVA_OPTS="$JAVA_OPTS ..."中添加如下命令：

-Dorg.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER=false，即 JAVA_OPTS="$JAVA_OPTS -Dorg.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER=false"，重启服务

备注：该参数默认值为 false。如果在启动脚本中未发现该参数，则不用添加。