# Tomcat安全

## 背景介绍

Tomcat是一个免费的开源的Serlvet容器，它是Apache基金会的Jakarta项目中的一个核心项目，由Apache，Sun和其它一些公司及个人共同开发而成。由于有了Sun的参与和支持，最新的Servlet和Jsp规范总能在Tomcat中得到体现。

尽管Tomcat的Web服务器功能较少，但是，Tomcat容易架设、容易移植、与其他服务器兼容性极好，相对于其它以C、C＋＋所编写的Web服务器，Tomcat比较能忍受远程缓冲区溢出的攻击。因为Tomcat的Java虚拟机是位于网络及操作系统之间，它可以防止几乎所有类型的缓冲区溢出攻击。此外，Tomcat的运行速度也比较快，足以运行今日大多数的企业网站。因此，Tomcat服务器的应用也是极为广泛的。

在使用Tomcat服务器时，我们要避免错误的配置引起的安全问题，同时，也使用更合理的配置或插件来加强Tomcat的安全防护。

eCity平台定制化的Tomcat是基于apache-tomcat-6.0.18进行修改，在***”..\工具\05 Tomcat安全”***中提供，修改内容都有批注，其它产品可以借鉴。

## Tomcat安全配置规范

Tomcat安全配置规范包括以下几部分

### 版本部署运行的策略

#### 及时更新补丁或升级系统，使用最新最稳定的安全版本

说明：系统的最原始攻击来自其本身的漏洞，因此一定要做好其漏洞的防护。搜索Tomcat最新最稳定的安全版本和安全补丁，下载并安装。

该规则主要针对发布版本，至于现网正在运行的版本，需要评估更新Tomcat版本对产品产生的影响，并酌情把握。

实施指导：

|  |
| --- |
| 下载最新最稳定的版本（以下提供的均为官方网址）   1. 在http://tomcat.apache.org/whichversion.html上，查看Servlet和JSP规范对应的Tomcat版本，以及各版本的详细说明，以方便选择最合适的版本。 2. 在http://tomcat.apache.org/download-\*\*.cgi上下载最新最稳定的安全版本。目前仅推荐60和55系列的版本，即将网址中的\*\*符号替换成60或者55。其他系列的版本已经归档，并且不再能得到支持。 |
| 安装步骤（以下皆以Tomcat6.0.20为例）   1. Window安装：   # 在http://tomcat.apache.org/download-60.cgi上下载apache-tomcat-6.0.20.exe程序  # 执行./apache-tomcat-6.0.20.exe  # 配置安装目录  # 进入Configuration配置界面  # 修改默认端口8080为其他不宜猜测的端口  # 修改默认管理员用户名admin  # 配置管理员密码为强口令  # 配置Java虚拟机路径为可用jre路径（建议版本在1.5以上）  # 进入安装等待状态   1. Unix安装   # 在http://tomcat.apache.org/download-60.cgi上，下载“Binary Distributions→Core”路径下的tar.gz  # gzip -d apache-tomcat-6.0.20.tar.gz  # tar -xvf apache-tomcat-6.0.20.tar  # cd ./apache-tomcat-6.0.20/bin  # ./startup.sh  如需配置管理员，可以在./conf/tomcat-user.xml文件中修改默认密码为强口令：  <tomcat-users>  <user name="admin" password="**XXXX**" roles="admin,manager" /> //修改XXXX为强口令  </tomcat-users> |
| 补丁  在安全报告http://tomcat.apache.org/security.html上，查找产品当前使用的版本所存在的安全漏洞，并根据安全报告中漏洞的威胁级别，以及产品本身的应用场景，选择下载补丁还是升级系统。  备注：一般不建议直接打补丁，建议升级到当前系列已被评估过的最新最稳定的安全版本。 |

#### 用于运行Tomcat的用户必须是一个没有特权的用户：既无权访问非必须的文件，又不能执行非必须的程序

说明：缺省情况下，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。

该安全配置项需要每个产品都要严格执行。

实施指导：

|  |
| --- |
| Window环境：   1. 新建普通用户tomcatuser，确定新用户tomcatuser不属于管理员组。 2. 设置权限：为tomcatuser用户添加Tomcat安装目录的读、写、执行的权限；为tomcatuser用户添加应用程序（例如webapps）目录的只读权限；如果某些应用程序需要写权限，请单独为其配置tomcatuser的写权限。 3. 打开“控制面板→管理工具→服务”，在本地服务中找到“Apache Tomcat”，打开其属性，在“登录”页签中配置帐户为tomcatuser。 4. 以tomcatuser用户重新登录后启动Tomcat服务器，或者在DOS窗口以RUNAS命令启用tomcatuser帐号，打开Tomcat服务。 |
| Linux环境：   1. 专门为Tomcat建立一个单独的用户和用户组。   # groupadd tomcatgroup  # useradd –g tomcatgroup tomcatuser –p tomcatuser   1. 设置权限：为tomcatuser用户添加Tomcat安装目录的读、写、执行的权限；为tomcatuser用户添加应用程序（例如webapps）目录的只读权限；如果某些应用程序需要写权限，请单独为其配置tomcatuser的写权限。 2. 以tomcatuser用户登录后启动Tomcat服务器。 |

#### 删除Tomcat默认管理控制台或管理员帐户

说明：默认情况下，Tomcat存在管理控制台，其地址一般为http://[IP]:[Port]/admin。管理台的应用文件，在$tomcat\server\webapps下，有admin和manager两个应用。其用户密码，在$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。

该安全配置项需要每个产品都要严格执行。

实施指导：

|  |
| --- |
| 方法一：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。   1. 删除$tomcat\webapps下admin.xml和manager.xml文件； 2. 删掉$tomcat\conf/tomcat-users.xml中的用户密码； 3. 删除$tomcat\server\webapps下的admin和manager两个应用； 4. 删除$tomcat\webapps下的admin和manager两个应用。 |
| 方法二：本方法适用于需要保留默认控制台和相关程序的应用场景。   1. 在$tomcat目录下新建一个myapps； 2. 在myapps下新建一个ROOT目录； 3. 修改配置文件$tomcat\conf\server.xml，将appBase的值修改为myapps：   **<Host name="localhost" appBase="myapps"** |

#### 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

说明：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。该安全配置项需要每个产品都要严格执行。

实施指导：

|  |
| --- |
| 1. 删除安装过程文件： //安装后删除   apache-tomcat-6.0.20.\* //以6.0.20版本为例   1. 删除非必需的管理应用和帮助应用：   $tomcat\webapss\\*  Tomcat6之前的版本的版本还需要删除以下目录：  $tomcat\server\webapps\\*   1. 删除系统示例程序和网页：   $tomcat\examples //安装时可以选择不安装这部分   1. 删除用户测试类的非必需文件。 |

#### 禁用应用程序自动部署功能

说明：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。

实施指导：

|  |
| --- |
| 修改配置文件$tomcat/conf/server.xml如下：  **autoDeploy="false"** //自动部署  **deployOnStartup="false"** //在启动时自动部署  备注：   1. 如果server.xml文件中不存在deployOnStartup参数，则必须手工添加，因为该参数的默认值是true。格式位置如下：   <Host …. **deployOnStartup="false"/**>   1. 关闭自动部署功能后，Tomcat不会自动发布$appBase目录下的web应用。对需要发布的Web应用，需要手工部署。手工部署的推荐方法，是在server.xml文件的Host标签内新建一个context，格式如下：   **<Context path="/myapp"** //虚拟路径  **reloadable="false"** //只在启动时加载该Web应用WEB-INF的lib和classes  **docBase="D:\myapp"** //应用程序的路径  **workDir="D:\myapp\work"**  //该web应用的工作目录  **/>** |

### 基本配置的安全策略

#### 禁用不必要的http方法：DELETE、PUT、TRACE等

说明：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。

在配置前，需要了解产品是否会使用到这些HTTP方法，如果用不到这些方法，则必须禁用。开放这些方法会被APPSCAN等工具扫描出来，并被评为中级安全风险。

实施指导：

|  |
| --- |
| 1. 禁用DELETE和PUT的方法：   在$tomcat/conf/web.xml检查readonly参数的值是否为true：  **<init-param>**  **<param-name>readonly</param-name>**  **<param-value>true</param-value>**  **</init-param>**  备注：如果不存在readonly参数，则不用配置，因为该参数的默认值为true；如果配置了该参数，则需要确保参数值为true。   1. 禁用Trace的方法：   在$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aloowTrace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>PUT</http-method>**  **<http-method>DELETE</http-method>**  **<http-method>TRACE</http-method>**  **</web-resource-collection>**  **<auth-constraint>**  **</auth-constraint>**  **</security-constraint>** |

#### 禁用webdav

说明：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。

该安全配置项需要每个产品都要严格执行。

实施指导：

|  |
| --- |
| 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>**  **...** |

#### 禁用Symbolic links

说明：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。

该安全配置项需要每个产品都要严格执行。

实施指导：

|  |
| --- |
| 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WEB-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 |

#### 定制Tomcat出错信息

说明：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。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。

定制化的错误文件可以参考”..\工具\05 Tomcat安全”目录中的error.htm文件。

实施指导：

|  |
| --- |
| 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>**  备注：可以在**error.htm**文件中自定义出错提示信息，**error.htm**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，禁止给响应吗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。 |

#### 关闭Tomcat的目录列表功能

说明：缺省情况下，访问tomcat下的一个web应用，如果输入是一个目录名，而且该目录下没有一个可用的welcome文件，那么tomcat会将该目录下的所有文件列出来，这种敏感信息泄露是严格禁止的。Tomcat5.0以后的版本出于安全考虑，默认的是关闭目录列表，但是为了程序调试方便，部分程序员也有可能开启这一功能。因此，要确认该项配置是否正确。

实施指导：

|  |
| --- |
| 在配置文件$tomcat/conf/web.xml中修改listing参数的值为false：  **<init-param>**  **<param-name>listings</param-name>**  **<param-value>false</param-value>**  **</init-param>** |

#### 更改Tomcat服务器默认端口

说明：Tomcat服务器提供一些默认端口，这些公开的端口很容易被黑客捕获利用，进而威胁到服务器和应用等。

需要注意产品的安装规划，安装规划中的端口规划不能为这些默认端口。

实施指导：

|  |
| --- |
| 修改$home/conf/server.xml文件，更改默认端口为其他，例如：   1. **<Server port="2324" shutdown="CLOSETHESERVER"> //默认端口8005** 2. **<Connector port="8108" protocol="HTTP/1.1" //默认端口8080**   **connectionTimeout="20000"**  **redirectPort="6743" /> //默认端口8443**  **3．<Connector port="3039" protocol="AJP/1.3" redirectPort="6743" /> //默认端口8009** |

#### 限制http请求的消息主体的大小

说明：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。在配置该属性前，需要了解业务是否有大数据量传输的可能性，如果有传输大数据的功能点，需要酌情提高消息主体的大小或者去掉该限制。

实施指导：

|  |
| --- |
| 在配置文件$tomcat/conf/server.xml中的每个***Connector***的“maxPostSize”属性为10240：  **<Connector port="8080" protocol="HTTP/1.1"**  **connectionTimeout="20000" maxPostSize="10240"**  **redirectPort="8443" />**  备注：推荐请求体大小限制为10240Byte，如果产品有特殊需求，可以进行相应调整。 |

#### 禁止配置Tomcat的网络连接超时时间为0或-1

说明：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。

实施指导：

|  |
| --- |
| 在配置文件$tomcat/conf/server.xml中的每个***Connector***的“connectionTimeout”属性为20000：  **<Connector port="8080" protocol="HTTP/1.1"**  **connectionTimeout="20000" maxPostSize="10240"**  **redirectPort="8443" />**  备注：如果产品有特殊性能需求，可以进行相应调整。 |

#### 配置Tomcat的会话超时时间为10分钟

说明：Tomcat服务器提供的默认超时时间为30分钟，这个时间过长，除了导致空闲连接消耗的处理资源损失外，受攻击的时间窗也相应变长。

在配置会话超时是时间时，也是充分考虑用户的易用性，当模块较多但超时时间较短的产品，会给用户造成频繁退出的现象，使用起来感受较差，如eCity平台各个模块的默认会话超时时间就不能设置过短。

实施指导：

|  |
| --- |
| 修改$tomcat下所有的web.xml配置文件如下：  **<session-config>**  **<session-timeout>10</session-timeout>** //将30修改为10  **</session-config>** |

#### 关闭会话回收功能

说明：如果每个请求都建立一个自己的façade，那么该功能可以进行详细的列表录入，但是，该功能也有可能将这些请求的信息泄露给没有创建façade的其他请求。

实施指导：

|  |
| --- |
| 在启动脚本$TOMCAT/bin/catalina.sh文件的启动分支语句中添加如下命令：  **-Dorg.apache.catalina.connector.RECYCLE\_FACADES=false**  备注：该参数仅出现在Tomcat6.0及其之后的版本中，且默认值为false。如果在启动脚本中未发现RECYCLE\_FACADES参数，可以不用添加。 |

#### 关闭crossContext功能

说明：当crossContext设置为true时，系统允许ServletContext.getContext调度另一个应用程序，这使得crossContext有创造恶意应用的可能，因此必须限制该功能。

实施指导：

|  |
| --- |
| 在启动脚本中添加如下命令：  **-Dorg.apache.catalina.connector.RECYCLE\_FACADES=false**  备注：该参数仅出现在Tomcat6.0及其之后的版本中，且默认值为false。如果在启动脚本中未发现RECYCLE\_FACADES参数，可以不用添加。 |

### 文件目录的控制权限

#### Tomcat根目录及其所有父目录只能由Tomcat属主用户修改（755）

说明：不仅文件本身，tomcat根目录及其父目录都必须只能由属主来改写

实施指导：

|  |
| --- |
| 如果要配置Tomcat根目录为/home/tomcat，首先要确认/、/home、/home/tomcat目录都只有Tomcat属主用户才可以修改，其次要使用Tomcat属主用户执行下列命令：  # chown 0 . bin conf  # chgrp 0 . bin conf  # chmod 755 . bin conf  备注：这里默认的属主是root用户。 |

#### 可执行文件只能由Tomcat属主用户修改（511）

说明：如果允许非属主用户对由属主执行或读写的文件有写权限，则会危及系统。比如，别人有可能会覆盖可执行文件或脚本，那么下一次启动时，就会执行恶意代码。

实施指导：

|  |
| --- |
| 设置可执行文件的权限，例如bin目录下的执行文件，使用Tomcat属主用户执行下列命令：  # cd $tomcat/bin  # chmod 511 \* |

#### 配置文件只能由Tomcat属主用户修改（700）

说明：别人有可能会覆盖配置文件，从而获取更高的权限或者更多的信息。

实施指导：

|  |
| --- |
| 设置可执行文件的权限，例如bin目录下的执行文件，使用Tomcat属主用户执行下列命令：  # cd $tomcat/conf  # chmod 700 \* |

#### 日志文件只能由Tomcat属主用户修改和执行（700）

说明：如果日志目录对非属主用户是可写的，别人就有可能用一个指向其他敏感文件的连接来覆盖日志文件，使那个文件被改写为杂乱的数据。如果日志文件本身对非属主用户是可写的，别人就可能伪造日志。

实施指导：

|  |
| --- |
| 使用Tomcat属主用户执行下列命令：  # cd $tomcat  # chmod 700 $tomcat/logs |

#### 只对管理员等少数用户开放的web服务，建议采用IP或IP段控制访问

说明：减少对外暴露的风险。

实施指导：

|  |
| --- |
| 修改$tomcat/conf/context.xml配置文件如下：  **<Valve className="org.apache.catalina.valves.RemoteHostValve"**  **allow="\*.mycompany.com,www.yourcompany.com" />**  **<Valve className="org.apache.catalina.valves.RemoteAddrValve"**  **allow=”192.168.5.\*"   deny="192.168.1.\*"/>**  备注：其中，**RemoteHostValve**过滤的是主机名，**RemoteAddrValve**过滤的是IP地址，多个ip或多个域名之间，可以使用“,”分隔。 |

#### 配置虚拟目录，用以隐藏后台路径

说明：隐藏后台路径，减少暴露给攻击者的威胁。

实施指导：

|  |
| --- |
| 修改$tomcat/conf/server.xml配置文件如下：  **<Context path="/javajia" docBase="webapps/lambes" debug="0" reloadable="true" > </Context>**  **<Context path="/java" docBase="f:/htdocs/demo/" debug="0" reloadable="true" > </Context>**  备注：其中path的值是虚拟目录，docbase若是以盘符开始，则是硬盘目录的绝对路径，否则便是相对于$tomcat的相对路径。 |

### SSI和CGI配置策略

#### 禁用SSI和CGI功能

说明： 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。

实施指导：

|  |
| --- |
| Tomcat默认不支持SSI和CGI，请确认配置文件$tomcat/conf/web.xml中，< servlet>标签下的SSI和CGI配置均处于xml注释状态。 |

#### 不允许使用SetUID程序，尤其是root身份的SetUID程序

说明：setuid是一个非常危险的命令，它使任意用户在运行指定程序时，拥有该程序属主的权限。因此会带来极大的系统漏洞问题。所以如非必要，不要使用SetUID程序，尤其是root身份的SetUID程序。

实施指导：

|  |
| --- |
| 1．找出使用SetUID的程序：  find / -perm -2000 -exec ls -l {} \;  2．确认这些程序是否必须使用SetUID，并且只能通过SetUID来实现该功能，如果不是，请取消其SetUID：  chmod –s programfile |

### 日志访问控制策略

#### 开启Tomcat的日志功能：正常的访问日志和错误请求日志。日志文件的记录中包含访问时间、内容、结果及请求用户的ip等关键信息

说明：这些日志可以提供异常访问的线索。Tomcat可以记录所有的访问请求，同样，异常的请求也会记录。日志文件可以记录系统发生的重要事件，可以帮助我们找到安全事件的原因等等，因此，日志文件必须包含访问的关键信息，例如请求用户的ip、访问的网址等。

注意，需要评估产品的功能是否会导致该类日志文件过大，如果过大，需要定期清理该日志，否则会造成服务器空间的急剧减小。

实施指导：

|  |
| --- |
| 在配置文件$tomcat/conf/server.xml中，在<Engine>标签里添加如下内容：  **<Valve className="org.apache.catalina.valves.AccessLogValve"**  **directory="logs" prefix="localhost\_access\_log." suffix=".txt"**  **pattern="common" resolveHosts="false"/>**  备注：该段信息默认存在<host>的标签的注释中，但是直接取消注释，并不能使其生效。例如Tomcat6.0.20便不支持除了<Engine>以外的标签。 |
| 在上文配置信息中的**pattern**参数，是用来配置日志基本格式的。  一般情况下，pattern可以直接设置成两种方式：common和combined，他们的真实值如下：  **common** ：%h %l %u %t %r %s %b //基本已经符合规则要求  **combined**：%h %l %u %t %r %s %b %{Referer}i %{User-Agent}i //增加了代理信息  参数值的具体含义如下：   |  |  | | --- | --- | | %a | 记录访问者的IP | | %A | 记录本地服务器的IP | | %b | 发送信息的字节数，不包括http头，如果字节数为0的话，显示为“-” | | %B | 发送信息的字节数，不包括http头 | | %h | 设置resolveHosts为false时，这里就是IP地址，否则就是服务器名称 | | %H | 访问者使用的协议，例如HTTP/1.1 | | %l | 登录标识中的逻辑用户名，通常返回值为“-” | | %m | 请求的HTTP方法（GET，POST等） | | %p | 访问的端口 | | %q | 查询字符串（即请求中?后面的部分） | | %r | 请求头，即请求的第一行（例如 POST / HTTP/1.1） | | %s | 响应请求的HTTP状态代码（例如 200，404等） | | %S | 用户的会话ID | | %t | 日期和时间 | | %u | 得到了验证的访问者，否则就是"-" | | %U | 请求的URL地址 | | %v | 服务器名称 | | %D | 请求响应时间，单位是毫秒 | | %T | 请求响应时间，单位是秒 | | %I | 当前请求的线程名称（可以在后期用来堆栈比较） | |

### 已知安全漏洞防护

#### 防护UTF-8导致的目录跨越漏洞

说明：该问题属于Tomcat5.5.27和Tomcat6.0.17之前的版本。漏洞的起因，在于Tomcat处理UTF-8编码时，没有正确转换，从而导致在处理包含%c0%ae%c0%ae的url时转换为类似../的形式，使得可以遍历系统任意文件，包括/etc/passwd等。

实施指导：

|  |
| --- |
| 方法一：为适应国际通用性，必须使用UTF-8的产品，建议通过升级修正该漏洞：   1. Tomcat5.5系列的版本，必须升级到5.5.27之后的稳定版本 2. Tomcat6.0系列的版本，必须升级到6.0.17之后的稳定版本 |
| 方法二：非必须应用UTF-8的产品，可以通过禁用UTF-8的方式规避该漏洞：   1. 打开配置文件$tomcat/conf/server.xml和$tomcat/conf/context.xml 2. 禁用**allowLinking** 3. 禁用**URIencoding**，或者设置**URIencoding**不等于**UTF8** |

### 安全模块及工具

#### 让Tomcat服务器在"监牢"中运行

说明： Linux内核的chroot机制。chroot是内核中的一个系统调用，软件可以通过调用库函数chroot，来更改某个进程所能见到的根目录。比如，一旦目录权限设定失误，被攻击的Tomcat进程可能会访问到Tomcat安装目录的父目录，甚至整个文件系统，因为Tomcat进程所处的根目录仍是整个文件系统的根。如果能够利用chroot将Tomcat限制在$tomcat内部，那么，Tomcat所能存取的文件都是$tomcat下的文件或其子目录下的文件。创建chroot"监牢"的作用就是将进程权限限制在文件系统目录树中的某一子树中。

实施指导：

|  |
| --- |
| 使用linux内核的chroot机制  这种方法的优点是无需进行另外的安装，直接使用root用户进行配置，缺点则是要将大量的、最终要用的各类库及相关配置文件等copy到虚拟根目录中。  1．以Tomcat用户登录，  # mkdir $chroot ///$chroot为希望建立的根目录  # cd $chroot  # mkdir etc lib bin sbin usr home //建立chroot环境必须具备的目录  # cd usr  # mkdir bin sbin lib local  # cd ..  # cp /bin/bash bin //将必须的程序、文件等拷贝到虚拟根目录中  # ldd /bin/bash //检查一下上面的bash程序用到哪些库  linux-gate.so.1 => (0xffffe000)  libreadline.so.4 => /lib/libreadline.so.4 (0x40036000)  libhistory.so.4 => /lib/libhistory.so.4 (0x40062000)  libncurses.so.5 => /lib/libncurses.so.5 (0x40069000)  libdl.so.2 => /lib/libdl.so.2 (0x400ae000)  libc.so.6 => /lib/tls/libc.so.6 (0x400b2000)  /lib/ld-linux.so.2 (0x40000000)  # cp /lib/{libreadline.so.4, libhistory.so.4, libncurses.so.5, libdl.so.2,ld-linux.so.2} lib  # cp /lib/tls/libc.so.6 lib //将这些库文件拷贝到虚根对应的目录中  如果需要用到其他的程序，可以像bash一样操作，将它们和它们的库文件拷贝到虚拟根目录的对应位置。  2．以root权限的用户登录，  # cd $chroot  # chroot . /bin/bash //启动监牢，在监牢中运行tomcat  但是具有root权限的用户，能够突破监牢，所以我们要采用降权之后的身份运行程序。   1. 使用降权用户   # cd $chroot  # cp /usr/bin/sudo ./usr/bin //采用步骤1中的方式，将sudo和它的库文件  ...... //复制到虚拟根目录  # cp /etc/passwd ./etc  # vi ./etc/passwd //修改虚根下的passwd文件，只保存用户tomcat  …… //或者修改伪造其他多余用户的信息  # sudo chroot . /bin/bash //如果不想使用sudo，也可以自己编写程序 |

### 其它设置

其它的Tomcat设置可能无助于安全防护，但是有助于WEB应用性能的提升。

|  |
| --- |
| 设置一：修改采用非阻塞IO接收请求，提高Tomcat并发处理效率。      1. 打开配置文件$tomcat/conf/server.xml，将启用端口的连接器协议按红字方式修改：  <Connector port="8031" protocol="org.apache.coyote.http11.Http11NioProtocol"                connectionTimeout="20000"                 redirectPort="8443" URIEncoding="utf-8"                 maxThreads="100" maxKeepAliveRequests="1"/> |
| 设置二： 打开Tomcat响应压缩功能，降低网络开销提高效率。  1. 打开配置文件$tomcat/conf/server.xml，将启用端口的连接器协议按红字方式修改：  <Connector port="8088" protocol="org.apache.coyote.http11.Http11NioProtocol"  connectionTimeout="20000" maxThreads="100"  compression="on" compressionableMimeType="text/html,text/xml,text/javascript,text/plain,applicatioin/json" redirectPort="7443" URIEncoding="UTF-8" allowTrace="false" /> |