# EPMS Tomcat安全加固配置手册

# 第1章 概述

## 1.1目标

现有的Web服务体系架构缺少有效的安全性支持，所以需要一个安全框架模型来解决Web服务中的各种安全问题。Web服务器是应用的载体，如果这个载体出现安全问题，那么运行在其中的Web应用程序的安全就得不到保障了。本文主要描述Apache Tomcat的安全加固和配置工作，最终用以指导系统实施。

## 1.2预期读者

本文档用于指导系统工程师进行系统实施工作，架构师和系统工程师应该通读本文档，选择适当方式用于自己的系统。

# 第2章 产品介绍

Tomcat是一个HTTP服务器，是Sun透过Java Community Process开发的、对广泛使用的Servlet和JavaServer Page(JSP)技术的正式参考实作。

Servlet和JSP技术用于建构HTTP服务器应用程序。虽然Servlet技术中加入了许多特性(包括存取安全性、Session管理和执行绪控制)。JSP技术提供了一种处理动态生成的 HTML 页面的简便方法，这些 HTML 页面被直接编译成 Servlet 以用于快速执行时作业。

Tomcat除了上述的两种技术保障安全之外，还可以通过配置Tomcat的参数以增加安全性。

# 第3章 Tomcat的安全加固配置

## 3.1 Tomcat版本统一

查看目前系统部署的Tomcat是否为统一的版本，切换到 tomcat的 \bin目录下：

epms@trust:~/tomcat/crmTomcat/bin> ./version.sh

Using CATALINA\_BASE: /home/epms/tomcat/crmTomcat

Using CATALINA\_HOME: /home/epms/tomcat/crmTomcat

Using CATALINA\_TMPDIR: /home/epms/tomcat/crmTomcat/temp

Using JRE\_HOME: /usr/local/java/jdk1.7.0\_79

Using CLASSPATH: /home/epms/tomcat/crmTomcat/bin/bootstrap.jar:/home/epms/tomcat/crmTomcat/bin/tomcat-juli.jar

Server version: Apache Tomcat/7.0.72

Server built: Sep 14 2016 12:12:26 UTC

Server number: 7.0.72.0

OS Name: Linux

OS Version: 3.0.76-0.11-default

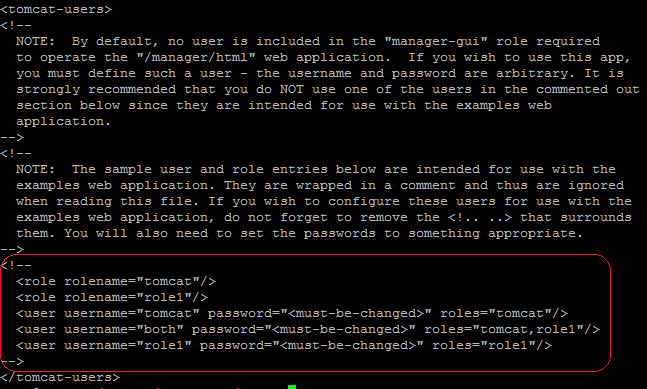
Architecture: amd64

JVM Version: 1.7.0\_79-b15

JVM Vendor: Oracle Corporation

**3.2 Tomcat帐号安全**

将 \conf\tomcat-users.xml中文件的所有用户都注释掉，配置如下：



如果业务需要内置的Tomcat用户，则需要给其分配合理的权限，密码设置符合密码策略，Tomcat角色类型如下：

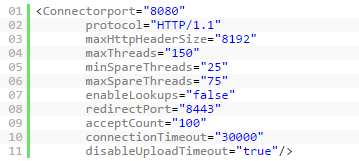
 role1： 具有读权限；

 tomcat： 具有读和运行权限；

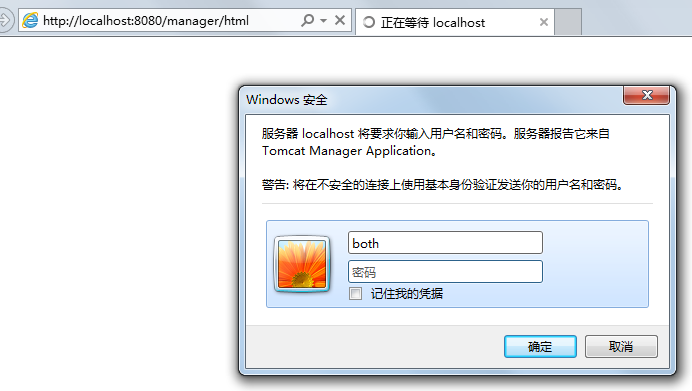
 admin： 具有读、运行和写权限；

 manager：具有远程管理权限。

如果使用启用了Tomcat用户，则需要设置Tomcat定时登出，将\conf\server.xml配置如下：



**加固检查：** 输入帐号和密码都无法登录



## 3.3除webapps应用

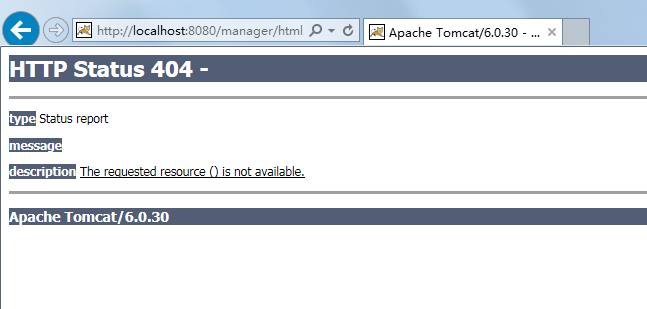
除了需要部署上去的应用，其余位于 \webapps文件夹中的应用如docs、examples、host-manager、manager和ROOT。

注：若无业务必要，请执行删除上述的应用包。

加固检查：

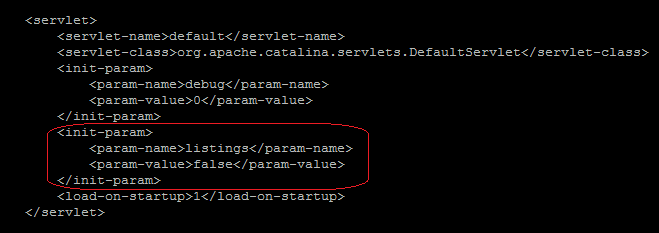
输入：http://localhost:8080/manager/html

出现的页面空白



## 3.4禁止Tomcat目录列表

确保 \conf\web.xml中listings的值为false：



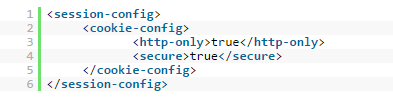
## ~~3.5设置Cookie的HttpOnly属性(此项不做)~~

~~因为目前框架中没有为Cookie添加HttpOnly属性，所以可以在Tomcat中设置。~~

~~第一步，在 \conf\context.xml文件中添加useHttpOnly="true"配置如下：~~

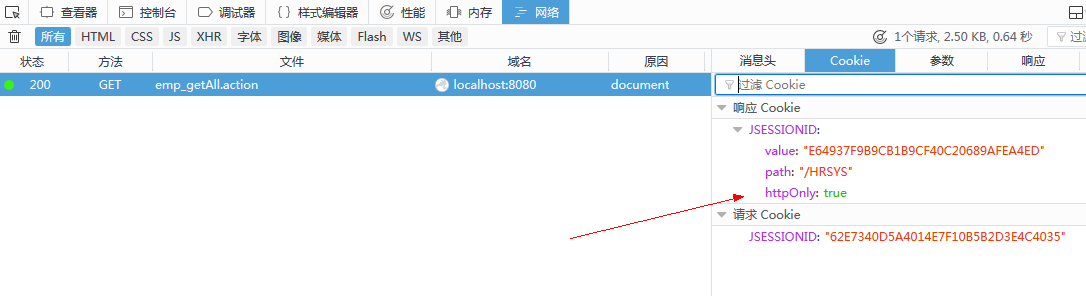
~~~~

~~第二步，在项目应用的web.xml中配置上：~~

~~~~

~~加固检查：~~

~~可以通过firebug查看cookie，出现HttpOnly属性~~

~~~~

## 3.6配置shutdown端口

在 \conf\server.xml中“<Server port="8005" shutdown="SHUTDOWN">”配置有允许任何人只要telnet到服务器的8005端口，输入"SHUTDOWN"，然后回车，服务器立即就被关掉了。从安全的角度上考虑，需要把这个shutdown指令改成一个别人不容易猜测的字符串。而且这个修改不影响shutdown.bat或shutdown.sh的执行。配置如下：

<Serverport="未被占用的端口" shutdown="较为复杂的字符串"> "EPMSHALT"

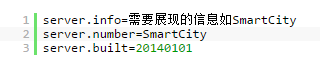
注：配置的端口需要大于1024。

加固检查：

使用telnet 127.0.0.1 8005进入，输入“SHUTDOWN”，Tomcat不会被关闭。

## 3.7隐藏Tomcat版本信息

在默认配置下，当应用出现异常时，客户端会显示Tomcat的版本信息。攻击者可以根据Tomcat版本信息选择漏洞库攻击，所以需要将Tomcat的版本信息隐藏，将 \lib\catalina.jar\org\apache\catalina\util中的配置ServerInfo.properties如下：



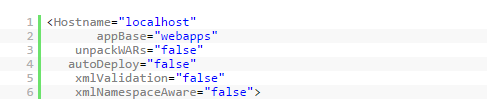
加固检测：

输入一个异常的URL看看出现的异常页面是否出现类似（无Tomcat的版本信息）如下内容：



## 3.8关闭war自动部署

默认的配置war放在 \webapps中会自动部署，所以关闭war自动部署防止被植入木马等恶意程序。将 \conf\server.xml配置如下：

     加固检查：

将一个war文件放进$CATALINA\_HOME\webapps，启动Tomcat后，war文件不会被部署。

## 3.9 Tomcat启动权限

|  |  |
| --- | --- |
| 08 | [dfhapp@localhost tomcat-demo]$ chmod 750 \* |
| 09 | [dfhapp@localhost tomcat-demo]$ chmod -R 540 bin/\* | |

|  |  |
| --- | --- |
| 10 | [dfhapp@localhost tomcat-demo]$ chmod -R 540 lib/\* |
| 11 | [dfhapp@localhost tomcat-demo]$ chmod -R 640 conf/\* | |

|  |  |  |  |
| --- | --- | --- | --- |
| 12 | | [dfhapp@localhost tomcat-demo]$ cd bin/ | |
| 01 | #启动Tomcat | |

|  |  |
| --- | --- |
| 02 | [dfhapp@localhost bin]$ ./startup.sh |
| 03 | UsingCATALINA\_BASE:   /soft/tomcat-demo | |

|  |  |
| --- | --- |
| 04 | UsingCATALINA\_HOME:   /soft/tomcat-demo |
| 05 | UsingCATALINA\_TMPDIR: /soft/tomcat-demo/temp | |

|  |  |
| --- | --- |
| 06 | UsingJRE\_HOME:        /usr |
| 07 | UsingCLASSPATH:      /soft/tomcat-demo/bin/bootstrap.jar | |

## 备注

## cookie的实现方案之一：

web.xml中配置：

<filter>

<filter-name>cookieFilter</filter-name>

<filter-class>com.icss.ebu.ami.portal.system.controller.CookieFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>cookieFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

package com.icss.ebu.ami.portal.system.controller;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import java.util.Date;

import java.util.Locale;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class CookieFilter implements Filter

{

@Override

public void init (FilterConfig filterConfig) throws ServletException

{

// TODO Auto-generated method stub

}

@Override

public void doFilter (ServletRequest request, ServletResponse response, FilterChain chain) throws IOException,

ServletException

{

HttpServletRequest req = (HttpServletRequest) request;

HttpServletResponse resp = (HttpServletResponse) response;

Cookie[] cookies = req.getCookies ();

if (cookies != null)

{

Cookie cookie = cookies[0];

if (cookie != null)

{

String value = cookie.getValue ();

StringBuilder builder = new StringBuilder ();

builder.append ("JSESSIONID=" + value + "; ");

builder.append ("Secure; ");

builder.append ("HttpOnly; ");

Calendar cal = Calendar.getInstance ();

cal.add (Calendar.HOUR, 1);

Date date = cal.getTime ();

Locale locale = Locale.CHINA;

SimpleDateFormat sdf = new SimpleDateFormat ("dd-MM-yyyy HH:mm:ss", locale);

builder.append ("Expires=" + sdf.format (date));

resp.setHeader ("Set-Cookie", builder.toString ());

}

}

chain.doFilter (req, resp);

}

@Override

public void destroy ()

{

// TODO Auto-generated method stub

}

}

# 第四章、