第 1 页 共 14 页

单点登录试验总结

一、 IBM Tivoli Access Manager概述 .......................................................... 2

二、 LTPA ........................................................................................................ 2

 三、 Global signon(GSO) ................................................................................ 3

 四、 HttpHeader .............................................................................................. 4

1. IBM Tivoli Access Manager概述

TAMeb主要结构是由WebSEAL、Policy Server、存储三者构成的。

TAMeb的结构原理为：

用户提交请求给WebSEAL，WebSEAL通过和存储以及认证数据库进行验证，然后对后台的资源进行提交，返回给客户。而TAMeb的主数据主要是存放在Policy Server上。

在XX项目中，我们主要应用的是TAMeb WebSEAL组件，其是TAM安全域中基于Web资源的资源安全性管理器。WebSEAL作为逆向Web代理，从web浏览器接收HTTP/HTTPS请求并交付来自其自己的web服务器或来自联结的后端的web应用程序服务器的内容。

WebSEAL提供以下功能：

支持多种认证方式

在XX项目的单点登录试验中，我们共采用了三种认证方式：

LTPA、GSO和Httpheader

接受Http和Https请求

通过WebSEAL集成和保护后端的服务器资源，管理访问控制。

作为逆向Web代理执行

对于客户端，WebSEAL可以作为Web服务器，而对于正在受保护的后端服务器，WebSEAL可作为Web浏览器

1. LTPA

LTPA全称Lightweight third party authentication。LTPA 定义了存储在客户端上的令牌格式。假如运行在不同机器上的第二个 WEB应用程序也使用 LTPA，服务器会检查用户是否已经通过认证。如果通过了，用户就不必再次提供证书。

TAM与Portal进行单点登录配置的主要步骤：

1. 从目标Portal的控制台，即Websphere的管理控制台中导出LTPA密钥，并将LTPA密钥复制到Portal服务器上。

2. 创建WebSEAL junction。   
以XX项目中的命令为例(下同)：

server task tamtest-webseald-testtam.hngytobacco.com create -t tcp -h icssprotal.hngytobacco.com -A -p 10038 -J trailer -F "c:\LTPA\_wmm.key" -Z "admin" -c all -f /portal

3. 导入用户或组

group import wpsadmins cn=wpsadmins,cn=groups,dc=hngytobacco,dc=com user import wpsadmin uid=wpsadmin,cn=users,dc=hngytobacco,dc=com user modify wpsadmin account-valid yes

4. 修改属性文件

修改\Tivoli\PDWeb\etc\webseald-default.conf文件。

将参数process-root-requests = always改为process-root-requests = never 重启TAM服务

5. 验证单点登录

访问：<http://tivoli.hngytobacco.com/portal/wps/myportal>

三、 Global signon(GSO)

单一注册解决方案，使用户能够向后端 Web 应用程序服务器提供用户名和密码。全局注册授予用户通过单个登录访问他们有权使用的计算资源的访问权。

创建GSO单点登录配置的主要步骤：

1. 创建resource和Junction

rsrc create rsrc\_oa\_gso

server task tamtest-webseald-testtam.hngytobacco.com create -t tcp -h

icssprotal.hngytobacco.com -p 10000 -S "C:\oa.fsso.conf" -f /oa

其中oa.fsso.conf的文件为Rone登录界面的信息

2. 导入用户为GSO账号

user import -gsouser icssadmin +%USER DN

3. 修改已有用户为GSO账户

user modify icssadmin gsouser yes

4. 创建GSO凭证

rsrccred create rsrc\_oa\_gso rsrcuser zhangl1001 rsrcpwd 1 rsrctype web user zhangl1001

5. 进行路径映射

将lib文件夹下的“jmt.conf”文件复制到/opt/pdweb/www-default/lib目录 server task default-webseald-tivoli.hngytobacco.com jmt load

6. 验证单点登录

访问：<http://tivoli.hngytobacco.com/oa/SignOnServlet>

四、 HttpHeader

思想：webseal建立了junction之后，在用户登录webseal成功，并试图访问后面应用的时候，会向后面的应用发送包，在包的头（header）里面会放入一些信息，一般放入用户名。login界面的改造就是要应用信任 webseal发过来的信息。

例子：TAM和Sync(同步工具)进行SSO。

思路：开发一个filter，添加到自身的Loginfilter之前，获取

HttpHeader的信息。验证是否为webseal的请求以及用户是否合法，验证成功，向session中注入相关信息，使该用户不需要再进行自身的验证。

Filter代码如下：

package hngy;

import java.io.IOException; import java.util.ArrayList; import java.util.Arrays;

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.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

/\*\*

 \* 获取到TAM发送过来的的用户id

 \* 判断是否为iv-user以及useid是否为合法用户  \* @author chenliang  \*  \*/

public class tamSSOFilter implements Filter {

private boolean CheckTrustHost = true;

private ArrayList list = new ArrayList();

public void destroy() {

 }

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

 throws IOException, ServletException {

HttpServletRequest httprequest = (HttpServletRequest)request;

HttpServletResponse httpresponse = (HttpServletResponse)response;

String userid = null;

/\*\*

   \* 获取webseal的服务器名称，如果为空，不处理    \*/

 if(httprequest.getHeader("iv\_server\_name") ==  null){

chain.doFilter(httprequest, httpresponse);    return;

  }

/\*\*

   \* 获取webseal发送的人员id，如果为空，不处理    \*/

userid = httprequest.getHeader("iv-user");

if (userid == null){

chain.doFilter(httprequest,httpresponse);

   return;

  }

 /\*\*

   \* 获取客户端请求的IP，如果未注册，不处理    \*/

  String remoteIp = request.getRemoteAddr();

if (CheckTrustHost && !list.contains(remoteIp)){

return;

  }

 /\*\*

   \* 向sesssion中注入相关参数，跳过业务系统的检查    \*/

  HttpSession session = httprequest.getSession();

 /\*\*

   \* 需要开发相关的接口，不同的业务系统方式不一样    \*/

   session.setAttribute("sync\_is\_login", Boolean.TRUE);

session.setAttribute("sync\_is\_admin", Boolean.TRUE);

chain.doFilter(httprequest, httpresponse);

}

//获取信任地址

public void init(FilterConfig config) throws ServletException {

String trusthost = config.getInitParameter("TrustHost");

//获取单点登录的方式

String ssotype = config.getInitParameter("SSOType");

if(null == trusthost||trusthost.equals("")||trusthost.length() ==0){

    CheckTrustHost = false;

 return

  }

  if(trusthost.indexOf(";")!= -1){

String hostValue[] = trusthost.split(";");

list.addAll(Arrays.asList(hostValue));

}

 }

}

Web.xml代码如下：

 <filter>

        <filter-name>tamSSOFilter</filter-name>

        <filter-class>hngy.tamSSOFilter</filter-class>

<init-param>

            <param-name>TrustHost</param-name>

            <param-value>10.158.157.214</param-value>

</init-param>

<init-param>

            <param-name>SSOType</param-name>

<param-value>httpheader</param-value>

 </init-param>

</filter>

<filter-mapping>

         <filter-name>tamSSOFilter</filter-name>

 <url-pattern>/app/\*</url-pattern>

</filter-mapping>