受影响系统

Apache Group Tomcat 7.0.0 - 7.0.29

Apache Group Tomcat 6.0.0 - 6.0.35

不受影响系统

漏洞简述

Apache Tomcat是一个流行的开放源码的JSP应用服务器程序。

Tomcat v7.0.30、6.0.36之前版本在FORM身份验证的实现上存在安全漏洞。在使用FORM验证时，若其他组件（如Single-Sign-On）在调用FormAuthenticator#authenticate()之前调用了request.setUserPrincipal()，则攻击者可以通过在URL结尾添加"/j\_security\_check"以绕过FORM验证。

漏洞利用方法

需要自己构造一个实现Single-Sign-On并且在调用FormAuthenticator#authenticate()之前调用了request.setUserPrincipal()的小型FORM验证系统

漏洞利用条件及利用方式

服务器或主机安装Tomcat版本为 6.0.0 - 6.0.35以及 7.0.0 -7.0.29,需要登陆认证。

漏洞利用情况

在安装Tomcat版本为 6.0.0 - 6.0.35以及 7.0.0 -7.0.29前提下，此漏洞利用方法是需要满足在使用FORM验证时存在调用FormAuthenticator#authenticate()之前调用了request.setUserPrincipal()，就可以通过在URL结尾添加"/j\_security\_check"以绕过FORM验证

修补情况

厂商补丁：

Apache Group

------------

目前厂商已经发布了升级补丁以修复这个安全问题，请到厂商的主页下载7.0.30和6.0.36或更高版本。

参考资料

[1] <http://seclists.org/bugtraq/2012/Dec/43>

[2] <http://www.securityfocus.com/bid/56812>

测试环境搭建

分析漏洞的实验环境：

操作系统: windows xp sp3

tomcat版本: 7.0.11

测试过程:

由于没有存在现成可用的POC，构造的单点登录页面没能利用这个漏洞，测试过程是在已有的页面后面添加/j\_security\_check

技术细节

漏洞概述:

在使用FORM验证时，若其他组件（如Single-Sign-On）在调用FormAuthenticator#authenticate()之前调用了request.setUserPrincipal()，则攻击者可以通过在URL结尾添加"/j\_security\_check"以绕过FORM验证，

分析过程:

由于构造的页面利用没有成功，这里仅对官方补丁来进行分析

|  |
| --- |
| --- tomcat/tc7.0.x/trunk/java/org/apache/catalina/realm/RealmBase.java 2012/08/27 22:27:31 1377891  +++ tomcat/tc7.0.x/trunk/java/org/apache/catalina/realm/RealmBase.java 2012/08/27 22:28:43 1377892  @@ -45,7 +45,6 @@  import org.apache.catalina.Wrapper;  import org.apache.catalina.connector.Request;  import org.apache.catalina.connector.Response;  -import org.apache.catalina.deploy.LoginConfig;  import org.apache.catalina.deploy.SecurityCollection;  import org.apache.catalina.deploy.SecurityConstraint;  import org.apache.catalina.mbeans.MBeanUtils;  @@ -822,31 +821,6 @@  if (constraints == null || constraints.length == 0)  return (true);    - // Specifically allow access to the form login and form error pages  - // and the "j\_security\_check" action  - LoginConfig config = context.getLoginConfig();  - if ((config != null) &&  - (Constants.FORM\_METHOD.equals(config.getAuthMethod()))) {  - String requestURI = request.getRequestPathMB().toString();  - String loginPage = config.getLoginPage();  - if (loginPage.equals(requestURI)) {  - if (log.isDebugEnabled())  - log.debug(" Allow access to login page " + loginPage);  - return (true);  - }  - String errorPage = config.getErrorPage();  - if (errorPage.equals(requestURI)) {  - if (log.isDebugEnabled())  - log.debug(" Allow access to error page " + errorPage);  - return (true);  - }  - if (requestURI.endsWith(Constants.FORM\_ACTION)) {  - if (log.isDebugEnabled())  - log.debug(" Allow access to username/password submission");  - return (true);  - }  - }  -  // Which user principal have we already authenticated?  Principal principal = request.getPrincipal();  boolean status = false; |

从补丁可以发现这里将专门允许访问form login和error page以及j\_security\_check的代码注释了。这里可以知道如果使return(true)成立就可以使登录成功。

根据if (requestURI.endsWith(Constants.FORM\_ACTION)) 查一下会发现FORM\_ACTION就是j\_security\_check

|  |
| --- |
| // Form based authentication constants  public static final String FORM\_ACTION = "/j\_security\_check";  public static final String FORM\_PASSWORD = "j\_password";  public static final String FORM\_USERNAME = "j\_username"; |

在满足URL以j\_security\_check结尾的条件下，就会返回true

在看下补丁处的函数hasResourcePermission是在域(Realm)中用来设置允许资源访问权限的。其中域(Realm)是用来进行权限管理的。hasResourcePermission会return <code>true </code>或者<code>false</code>

|  |
| --- |
| /\*\*  \* Perform access control based on the specified authorization constraint.  \* Return <code>true</code> if this constraint is satisfied and processing  \* should continue, or <code>false</code> otherwise.  \*  \* @param request Request we are processing  \* @param response Response we are creating  \* @param constraints Security constraint we are enforcing  \* @param context The Context to which client of this class is attached.  \*  \* @exception IOException if an input/output error occurs  \*/  @Override  public boolean hasResourcePermission(Request request,  Response response,  SecurityConstraint []constraints,  Context context) |

结合漏洞公告说明，在使用FORM验证时，若其他组件（如Single-Sign-On）在调用FormAuthenticator#authenticate()之前调用了request.setUserPrincipal()，此处单点登录sso调用了request.setUserPrincipal()并且sso是在FormAuthenticator#authenticate()之前的，满足条件

查看sso处调用request.setUserPrincipal

|  |
| --- |
| SingleSignOnEntry entry = lookup(cookie.getValue());  if (entry != null) {  if (containerLog.isDebugEnabled())  containerLog.debug(" Found cached principal '" +  (entry.getPrincipal() != null ? entry.getPrincipal().getName() : "") + "' with auth type '" +  entry.getAuthType() + "'");  request.setNote(Constants.REQ\_SSOID\_NOTE, cookie.getValue());  // Only set security elements if reauthentication is not required  if (!getRequireReauthentication()) {  request.setAuthType(entry.getAuthType());  request.setUserPrincipal(entry.getPrincipal()); //关注此处的getPrincipal  }  } else {  if (containerLog.isDebugEnabled())  containerLog.debug(" No cached principal found, erasing SSO cookie");  cookie.setMaxAge(0);  response.addCookie(cookie);  } |

查看下entry.getPrincipal()的代码

|  |
| --- |
| /\*\*  \* Gets the <code>Principal</code> that has been authenticated by  \* the SSO.  \*/  public Principal getPrincipal() {  return (this.principal);  } |

会发现getPrincipal()回去获取<code>Principal</code>此处的值，结合上面分析的hasResourcePermission可以得知此处如果返回true，则sso处会设置权限从而绕过FORM验证

总结: 此处分析的是单点登录sso的情况，若在在URL结尾添加"/j\_security\_check"，则hasResourcePermission会返回使<code>true</code>

进而再调用FormAuthenticator#authenticate()之前调用了request.setUserPrincipal()会设置Principal为true，进而绕过FROM验证