

# CVE-2022-0540\_Jira 身份验证绕过漏洞

## 前言

在上篇分析 [CVE-2022-26135Atlassian Jira Mobile Plugin SSRF 漏洞](#) 之后，发现在此之前，jira 也曾爆出过身份验证绕过漏洞，CVE 编号为 cve-2022-0540。趁着环境还热乎，对其产生的原理和代码进行一波分析和学习。

## 漏洞描述

Atlassian Jira 是澳大利亚 Atlassian 公司的一套缺陷跟踪管理系统。该系统主要用于对工作中各类问题、缺陷进行跟踪管理。攻击者可利用此漏洞向目标系统发送特制的 HTTP 请求，以使用受影响的配置绕过 WebWork 操作中的身份验证和授权要求。

## 利用范围

### Jira

Jira 所有版本 < 8.13.18

Jira 8.14.x、8.15.x、8.16.x、8.17.x、8.18.x、8.19.x

Jira 8.20.x < 8.20.6

Jira 8.21.x

### Jira Service Management

Jira Service Management 所有版本 < 4.13.18

Jira Service Management 4.14.x、4.15.x、4.16.x、4.17.x、4.18.x、4.19.x

Jira Service Management 4.20.x < 4.20.6

Jira Service Management 4.21.x

## 漏洞分析

## 环境搭建

本次环境使用 docker 搭建

## Dockerfile

```
Plaintext
FROM atlassian/jira-software:8.13.17

USER root

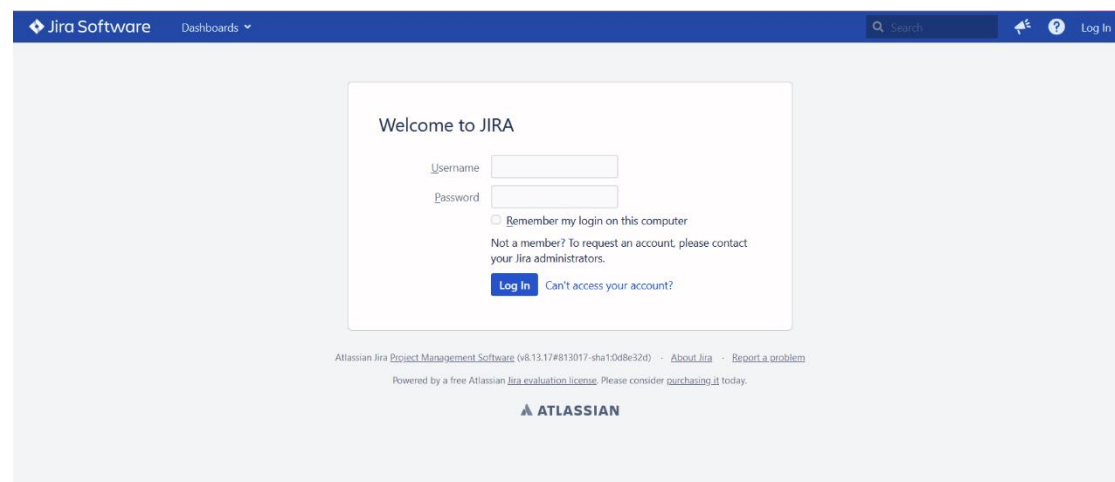
COPY "atlassian-agent.jar" /opt/atlassian/jira/

RUN echo 'export CATALINA_OPTS="-
javaagent:/opt/atlassian/jira/atlassian-agent.jar -
agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=*:5005
${CATALINA_OPTS}"' >> /opt/atlassian/jira/bin/setenv.sh
```

5005 为 idea debug 端口；atlassian-agent.jar 项目地址：

<https://github.com/hgqapp/atlassian-agent>，使用 maven 编译为 jar 文件即可。

后续创建容器根据提示配置即可。



调试源码将以下三个文件夹设置为 Libraries 即可

```
Plaintext
\atlassian-jira-software-8.13.17-standalone\lib
\atlassian-jira-software-8.13.17-standalone\atlassian-jira\WEB-INF\classes
\atlassian-jira-software-8.13.17-standalone\atlassian-jira\WEB-INF\lib
```

## 前置知识

在漏洞分析之前，先简单了解一下 Jira 相关的背景知识

## WebWork

Jira 使用 MVC 框架 WebWork 来处理用户发起的 WEB 请求。每个请求都是使用 WebWork action 来处理，在其中又使用了其他的 utility and Manager classes 来完成一个任务。作为响应返回给客户端的 HTML 大部分都是 View 层的 JSP 生成的。URL 中的“.jspx”后缀标识其后端对应的是一个 JSP 文件。

## Seraph

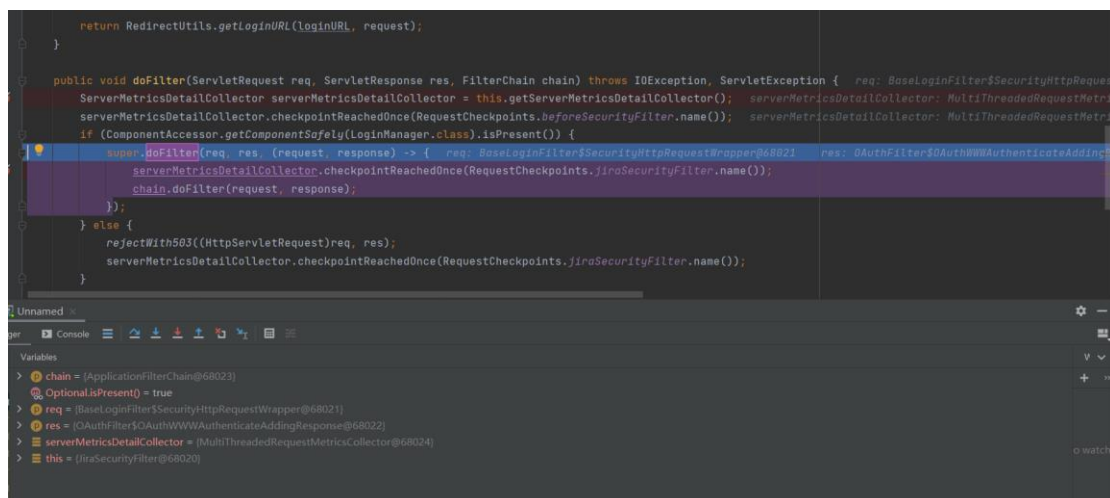
Seraph 是一个开源认证框架，主要由 Atlassian 开发和维护。Jira、Confluence 的登录认证是都由 Seraph 来负责的。Seraph 是通过 Servlet 的 Filter 实现的。Seraph 的功能只是用来在给定一个 Web 请求的情况下，将该请求与特定用户相关联。

## 动态分析

通过前置知识可以了解到 Seraph 是一个开源认证框架，也是 jira 核心身份验证机制。Seraph 是通过 Servlet 和 Filter 实现的。

查看 Filter，发现 com.atlassian.jira.security.JiraSecurityFilter.class

在 doFilter 中调用父方法 super.doFilter



```
return RedirectUtils.getLoginURL(loginURL, request);
}

public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException {
    req: BaseLoginFilter$SecurityHttpRequestWrapper@68021; res: OAuthFilter$OAuthWWWAuthenticateAddingResponse@68022;
    ServerMetricsDetailCollector serverMetricsDetailCollector = this.getServerMetricsDetailCollector(); serverMetricsDetailCollector: MultiThreadedRequestMetricsCollector@68023;
    serverMetricsDetailCollector.checkpointReachedOnce(RequestCheckpoints.beforeSecurityFilter.name()); serverMetricsDetailCollector: MultiThreadedRequestMetricsCollector@68023;
    if (ComponentAccessor.getComponentSafely(LoginManager.class).isPresent()) {
        super.doFilter(req, res, (request, response) -> { req: BaseLoginFilter$SecurityHttpRequestWrapper@68021; res: OAuthFilter$OAuthWWWAuthenticateAddingResponse@68022;
            serverMetricsDetailCollector.checkpointReachedOnce(RequestCheckpoints.jiraSecurityFilter.name());
            chain.doFilter(request, response);
        });
    } else {
        rejectWith503((HttpServletRequest)req, res);
        serverMetricsDetailCollector.checkpointReachedOnce(RequestCheckpoints.jiraSecurityFilter.name());
    }
}
```

Variables

- > chain = (ApplicationFilterChain@68023)
- > Optional.isPresent() = true
- > req = (BaseLoginFilter\$SecurityHttpRequestWrapper@68021)
- > res = (OAuthFilter\$OAuthWWWAuthenticateAddingResponse@68022)
- > serverMetricsDetailCollector = (MultiThreadedRequestMetricsCollector@68024)
- > this = (JiraSecurityFilter@68020)

com.atlassian.seaph.filter.SecurityFilter#doFilter

这里处于 atlassian-seaph-4.0.4.jar 中 filter 也就是 seraph 过滤器，它会根据请求用户

权限进行判断，进一步确定所需要的角色，确定是否需要认证。

```
public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException {
    log.debug("doFilter : Storing the originally requested URL (atlassian.core.seraph.original.url=" + originalURL + ")");
    log.debug("doFilter : Storing the originally requested URL (atlassian.core.seraph.original.url=" + originalURL + ")");

    req.setCharacterEncoding("UTF-8");
    HttpServletResponse res = (HttpServletResponse) res;
    res.setContentType("text/html; charset=UTF-8");

    String originalURL = httpServletRequest.getRequestURL().toString();
    if (originalURL.startsWith("/secure/")) {
        log.debug("doFilter : Storing the originally requested URL (atlassian.core.seraph.original.url=" + originalURL + ")");
    }

    Set<String> requiredRoles = new HashSet();
    Set<String> missingRoles = new HashSet();
    Iterator var11 = this.getSecurityConfig().getServices().iterator();

    while(var11.hasNext()) {
        SecurityService service = (SecurityService)var11.next();
        Set<String> serviceRoles = service.getRequiredRoles(httpServletRequest);
        requiredRoles.addAll(serviceRoles);
    }

    if (dbg) {
        log.debug("doFilter : requiredRoles = " + requiredRoles);
    }

    boolean needsAuth = false;
    Authenticator authenticator = this.getSecurityConfig().getAuthenticator();
    Principal user = authenticator.getUser(httpServletRequest, httpServletResponse);
    if (user == null) {
        AuthType authType = AuthType.getAuthType(httpServletRequest, this.getSecurityConfig());
        if (authType != null) {
            if (authType == AuthType.CONTEXT && httpServletRequest.getSession() != null) {
                httpServletResponse.sendError(401, "No " + authType + " cookie, but no valid cookie was sent.");
                return;
            }
        }
    }
}
```

持续跟进，发现会通过循环会出现三种 Service

## JiraPathService

```
if (dbg) {
    log.debug("doFilter : Storing the originally requested URL (atlassian.core.seraph.original.url=" + originalURL + ")");
}

Set<String> requiredRoles = new HashSet();
Set<String> missingRoles = new HashSet();
Iterator var11 = this.getSecurityConfig().getServices().iterator();

while(var11.hasNext()) {
    SecurityService service = (SecurityService)var11.next();
    Set<String> serviceRoles = service.getRequiredRoles(httpServletRequest);
    requiredRoles.addAll(serviceRoles);
}

if (dbg) {
    log.debug("doFilter : requiredRoles = " + requiredRoles);
}
```

Variables

- chain = (JiraSecurityFilter\$lambda@44448)
- collections\$UnmodifiableCollection\$1.next() = (JiraPathService@44467)
- dbg = false
- httpServletRequest = (BaseLoginFilter\$SecurityHttpRequestWrapper@44424)
- httpServletResponse = (OAuthFilter\$OAuthWWWAuthenticateAddingResponse@44425)
- METHOD = "doFilter"
- missingRoles = [HashSet@44461] []
- originalURL = "/download/batch/com.atlassian.jira.jira-tzdetect-plugin:tzdetect-lib/com.atlassian.jira.jira-tzdetect-plugin:tzdetect-lib.js"
- req = (BaseLoginFilter\$SecurityHttpRequestWrapper@44424)
- requiredRoles = [HashSet@44459] []
- res = (OAuthFilter\$OAuthWWWAuthenticateAddingResponse@44425)
- service = (JiraPathService@44467)
- this = (JiraSecurityFilter@44423)

JiraPathService 是处理：如果请求的 servlet 路径以 /secure/admin/ 开头，那其角色权限必须是 admin（管理员权限）。

```
package com.atlassian.jira.security;

import ...

public class JiraPathService implements SecurityService {
    private static final String SECURE_ADMIN_PREFIX = "/secure/admin/";
    private static final Set<String> ADMIN = ImmutableSet.of("admin");

    public JiraPathService() {
    }

    public void init(Map<String, String> stringStringMap, SecurityConfig securityConfig) {
    }

    public void destroy() {
    }

    public Set<String> getRequiredRoles(HttpServletRequest httpServletRequest) {
        String servletPath = httpServletRequest.getServletPath();
        return (Set)(servletPath != null && servletPath.startsWith("/secure/admin/") ? ADMIN : ImmutableSet.of());
    }
}

Variables
> httpServletRequest = (BaseLoginFilter$SecurityHttpRequestWrapper@44937)
> this = (JiraPathService@44960)
```

## WebworkService

```
Set<String> requiredRoles = new HashSet(); requiredRoles: "[]"
Set<String> missingRoles = new HashSet(); missingRoles: "[]"
Iterator var11 = this.getSecurityConfig().getServices().iterator();

while(var11.hasNext()) {
    SecurityService service = (SecurityService)var11.next(); service: WebworkService@44481
    Set<String> serviceRoles = service.getRequiredRoles(httpServletRequest);
    requiredRoles.addAll(serviceRoles);
}

if (dbg) {
    log.debug("doFilter : requiredRoles = " + requiredRoles);
}

Variables
> chain = (JiraSecurityFilter$lambda@44448)
> Collections$UnmodifiableCollection$1.next() = (WebworkService@44481)
> dbg = false
> httpServletRequest = (BaseLoginFilter$SecurityHttpRequestWrapper@44424)
> httpServletResponse = (OAuthFilter$OAuthWWWAuthenticateAddingResponse@44425)
> METHOD = "doFilter: "
> missingRoles = (HashSet@44461) "[]"
> originalURL = "/download/batch/com.atlassian.jira.jira-tzdetect-plugin:tzdetect-lib/com.atlassian.jira.jira-tzdetect-plugin:tzdetect-lib.js"
> req = (BaseLoginFilter$SecurityHttpRequestWrapper@44424)
> requiredRoles = (HashSet@44459) "[]"
> res = (OAuthFilter$OAuthWWWAuthenticateAddingResponse@44425)
> service = (WebworkService@44481)
> this = (JiraSecurityFilter@44423)
```

WebworkService 则是在 actions.xml 文件中获取角色所需要的 webwork 配置

```
public WebworkService() {
}

public void init(Map<String, String> params, SecurityConfig config) {
    try {
        String extension = (String)params.get("action.extension");
        if (extension == null) {
            extension = "action";
        }

        String actionsXmlFile = (String)params.get("actions.xml.file");
        if (actionsXmlFile == null) {
            actionsXmlFile = "actions";
        }

        this.configureActionMapper(extension, actionsXmlFile);
    } catch (RuntimeException var5) {
        log.error("Failed to initialise WebworkService", var5);
    }
}

private void configureActionMapper(String extension, String actionsXmlFile) {
    Document doc = this.parseActionsXmlFile(actionsXmlFile);
    NodeList actions = doc.getElementsByTagName("action");
    String rootRolesRequired = this.overrideRoles((String)null, doc.getDocumentElement());
    Map<String, String> pathMaps = new HashMap();

    for(int i = 0; i < actions.getLength(); ++i) {
        Element action = (Element)actions.item(i);
        String actionName = action.getAttribute("name");
        String actionAlias = action.getAttribute("alias");
        String actionRolesRequired = this.overrideRoles(rootRolesRequired, action);
        if (actionRolesRequired != null) {
            if (actionAlias != null) {
                pathMaps.put(actionAlias, "/" + actionAlias + "." + extension);
                this.rolesMap.put(actionAlias, actionRolesRequired);
                pathMaps.put(actionAlias + "!*", "/" + actionAlias + ".*" + extension);
                this.rolesMap.put(actionAlias + "!", actionRolesRequired);
            }
        }
    }
}

Variables
> Collections$UnmodifiableCollection$1.next() = (WebworkService@44967)
> request = (BaseLoginFilter$SecurityHttpRequestWrapper@44937)
> this = (WebworkService@44967)
```

经过调试，会多次进入 `getRequiredRoles` 函数，其中获取 URI 的方式为 `getRequestURL`

```
134 public Set<String> getRequiredRoles(HttpServletRequest request) { request: BaseLoginFilter$SecurityHttpRequestWrapper@44945
135     Set<String> requiredRoles = new HashSet(); requiredRoles: "[]"
136     String currentURL = request.getRequestURL(); request: BaseLoginFilter$SecurityHttpRequestWrapper@44945
137     int lastSlash = currentURL.lastIndexOf(47);
138     String targetURL;
139     if (lastSlash > -1) {
140         targetURL = currentURL.substring(lastSlash);
141     } else {
142         targetURL = currentURL;
143     }
144
145     String actionMatch = this.actionMapper.get(targetURL);
146     if (actionMatch != null) {
147         String rolesStr = (String)this.rolesMap.get(actionMatch);
148         StringTokenizer st = new StringTokenizer(rolesStr, ",");
149         while(st.hasMoreTokens()) {
150             requiredRoles.add(st.nextToken());
151         }
152     }
153     return Collections.unmodifiableSet(requiredRoles);
154 }
155
156 named <

Console
> HttpServletRequestWrapper.getRequestURL() = "/secure/insightPluginShowGeneralConfiguration.jsps"
> request = (BaseLoginFilter$SecurityHttpRequestWrapper@44945)
> requiredRoles = (HashSet@44977) "[]"
> this = (WebworkService@44966)
```

继续跟进发现在通过 `getRequestURL` 方式提取请求 URL 后，会通过提取最后一个/后面的接口产生一个 `targetURL`，这里传入的是 `/secure/InsightPluginShowGeneralConfiguration.jsps`，而 `targetURL` 为 `/InsightPluginShowGeneralConfiguration.jsps`

```
154 public Set<String> getRequiredRoles(HttpServletRequest request) { request: BaseLoginFilter$SecurityHttpRequestWrapper@444945
155     Set<String> requiredRoles = new HashSet(); requiredRoles: "[]"
156     String currentURL = request.getRequestURI(); currentURL: "/secure/InsightPluginShowGeneralConfiguration.jspa" request: BaseLoginFilter$SecurityHttpRequestWrapper@444945
157     int lastSlash = currentURL.lastIndexOf(47); lastSlash: 7
158     String targetURL; targetURL: "/InsightPluginShowGeneralConfiguration.jspa"
159     if (lastSlash > -1) {
160         targetURL = currentURL.substring(lastSlash); lastSlash: 7
161     } else {
162         targetURL = currentURL; currentURL: "/secure/InsightPluginShowGeneralConfiguration.jspa"
163     }
164
165     String actionMatch = this.actionMapper.get(targetURL); targetURL: "/InsightPluginShowGeneralConfiguration.jspa"
166     if (actionMatch != null) {
167         String rolesStr = (String)this.rolesMap.get(actionMatch);
168         StringTokenizer st = new StringTokenizer(rolesStr, ", ");
169         while(st.hasMoreTokens()) {
170             requiredRoles.add(st.nextToken());
171         }
172     }
173
174     return Collections.unmodifiableSet(requiredRoles);
175 }
```

Variables

- currentURL = "/secure/InsightPluginShowGeneralConfiguration.jspa"
- HttpServletRequestWrapper.getRequestURI() = "/secure/InsightPluginShowGeneralConfiguration.jspa"
- lastSlash = 7
- request = (BaseLoginFilter\$SecurityHttpRequestWrapper@444945)
- requiredRoles = (HashSet@444777) []
- targetURL = "/InsightPluginShowGeneralConfiguration.jspa"
- this = (WebSecurityService@444916)
- oothisactionMapper = (ConcurrentHashMap@444981) "Mappings: /admin/roles/EditUserProjectRoles.jspa=admin/roles/EditUserProjectRoles.jspa/AddNewIssueTypeToScheme.jspa=AddNewIssueTypeToScheme.jspa/TranslateCustomField.jspa=TranslateCustomField.jspa/DeleteUserProjectRoles.jspa=DeleteUserProjectRoles.jspa"
- oothisrolesMap = (ConcurrentHashMap@444980) Unable to evaluate the expression Cannot find source class for java.util.Map

## JiraSeraphSecurityService

```
}
Set<String> requiredRoles = new HashSet(); requiredRoles: "[]"
Set<String> missingRoles = new HashSet(); missingRoles: "[]"
Iterator var11 = this.getSecurityConfig().getServices().iterator();

while(var11.hasNext()) {
    SecurityService service = (SecurityService)var11.next(); service: JiraSeraphSecurityService@444494
    Set<String> serviceRoles = service.getRequiredRoles(httpServletRequest); httpServletRequest: BaseLoginFilter$SecurityHttpRequestWrapper@44424
    requiredRoles.addAll(serviceRoles);
}

if (dbg) {
    Log.debug("#doFilter==requiredRoles = " + requiredRoles);
}
```

Variables

- chain = (JiraSecurityFilter\$lambda@44448)
- dbg = false
- HttpServletRequest = (BaseLoginFilter\$SecurityHttpRequestWrapper@44424)
- HttpServletResponse = (OAuthFilter\$OAuthWWWAuthenticateAddingResponse@44425)
- METHOD = "doFilter"
- missingRoles = (HashSet@44461) []
- originalURL = "/download/batch/com.atlassian.jira.jira-tzdetect-plugin:tzdetect-lib/com.atlassian.jira.jira-tzdetect-plugin:tzdetect-lib.js"
- req = (BaseLoginFilter\$SecurityHttpRequestWrapper@44424)
- requiredRoles = (HashSet@44459) []
- res = (OAuthFilter\$OAuthWWWAuthenticateAddingResponse@44425)
- service = (JiraSeraphSecurityService@444494)
- this = (JiraSecurityFilter@44423)

继续往下就是第三个服务 JiraSeraphSecurityService，作用是在所有插件的 atlassian-plugin.xml 文件中获取角色所需的 webwork 操作配置

```
1  // ...
2
3  package com.atlassian.jira.plugin.webwork;
4
5  import ...
6
7  public class JiraSeraphSecurityService implements SecurityService {
8      private static final Logger LOG = LoggerFactory.getLogger(JiraSeraphSecurityService.class);
9
10     public JiraSeraphSecurityService() {
11     }
12
13     public void init(Map<String, String> params, SecurityConfig config) {
14     }
15
16     public void destroy() {
17     }
18
19     public Set<String> getRequiredRoles(HttpServletRequest request) { request: BaseLoginFilter$SecurityHttpRequestWrapper@45062
20     }
21
22     return (Set)ComponentAccessor.getComponentSafely(LoginManager.class).map((LoginManager) -> {
23         return loginManager.getRequiredRoles(request);
24     }).orElseThrow(() -> {
25         LOG.debug("Request {} bypassed Johnson but still requested a security check", request.getRequestURI());
26         return new IllegalStateException("Still initializing");
27     });
28 }
29 }
```

Dynamic

Console

Variables

- request = (BaseLoginFilter\$SecurityHttpRequestWrapper@45062)
- this = (JiraSeraphSecurityService@45110)
- WebworkService.getRequiredRoles(HttpServletRequest) = (Collections\$UnmodifiableSet@45109) Unable to evaluate the expression Cannot find source class for java.util.Collection

在跟进 JiraSeraphSecurityService 时，发现会调用 WebworkPluginSecurityServiceHelper.getRequiredRoles，和 WebworkService.getRequiredRoles 代码是相同的

```
WebworkPluginSecurityServiceHelper.class
Decompiled .class file, bytecode version: 52.0 (Java 8)

JiraSeraphSecurityService
90 }
91
92 public Set<String> getRequiredRoles(HttpServletRequest request) { request: BaseLoginFilter$SecurityHttpRequestWrapper@45062
93     Set<String> requiredRoles = new HashSet();
94     String currentURL = request.getRequestURI();
95     int lastSlash = currentURL.lastIndexOf(47);
96     String targetURL;
97     if (lastSlash > -1) {
98         targetURL = currentURL.substring(lastSlash);
99     } else {
100         targetURL = currentURL;
101     }
102
103     String actionMatch = this.actionMapper.get(targetURL);
104     if (actionMatch != null) {
105         String rolesStr = (String)this.rolesMap.get(actionMatch);
106         StringTokenizer st = new StringTokenizer(rolesStr, ", ");
107
108         while(st.hasMoreTokens()) {
109             requiredRoles.add(st.nextToken());
110         }
111     }
112
113     return Collections.unmodifiableSet(requiredRoles);
114 }
115
116 public void start() throws Exception {
117     this.eventPublisher.register(this);
118     this.generatePathMaps();
119 }
```

Dynamic

Console

Variables

- Invokers\$Holder.linkToTargetMethod(Object, Object) = (JiraSeraphSecurityService\$lambda@45136)
- request = (BaseLoginFilter\$SecurityHttpRequestWrapper@45062)
- this = (WebworkPluginSecurityServiceHelper@45137)



接口权限必须是 admin

```
91
92 public Set<String> getRequiredRoles(HttpServletRequest request) { request: BaseLoginFilter$SecurityHttpRequestWrapper@45067
93 Set<String> requiredRoles = new HashSet(); requiredRoles: "[admin]"
94 String currentURL = request.getRequestURI(); request: BaseLoginFilter$SecurityHttpRequestWrapper@45067 currentURL: "/secure/insightPluginShowGeneralConfiguration.jspa"
95 int lastSlash = currentURL.lastIndexOf(47); lastSlash: 7
96 String targetURL; targetURL: "/insightPluginShowGeneralConfiguration.jspa"
97 if (lastSlash > -1) {
98     targetURL = currentURL.substring(lastSlash); lastSlash: 7
99 } else {
100     targetURL = currentURL; currentURL: "/secure/insightPluginShowGeneralConfiguration.jspa"
101 }
102
103 String actionMatch = this.actionMapper.get(targetURL); targetURL: "/insightPluginShowGeneralConfiguration.jspa" actionMatch: "insightPluginShowGeneralConfiguration"
104 if (actionMatch != null) {
105     String rolesStr = (String)this.rolesMap.get(actionMatch); actionMatch: "insightPluginShowGeneralConfiguration"
106     StringTokenizer st = new StringTokenizer(rolesStr, ",");
107
108     while(st.hasMoreTokens()) {
109         requiredRoles.add(st.nextToken());
110     }
111 }
112
113 return Collections.unmodifiableSet(requiredRoles); requiredRoles: "[admin]"
114 }
115
116 public void start() throws Exception {
117     this.eventPublisher.register( @ this);
118     this.generatePathMaps();
119 }
120
121 }
```

Variables

- actionMatch = "insightPluginShowGeneralConfiguration"
- currentURL = "/secure/insightPluginShowGeneralConfiguration.jspa"
- lastSlash = 7
- request = BaseLoginFilter\$SecurityHttpRequestWrapper@45067
- requiredRoles = [HashSet@45114] "[admin]"
- StringTokenizer.hasMoreTokens() = false
- targetURL = "/insightPluginShowGeneralConfiguration.jspa"
- this = (WebworkPluginSecurityServiceHelper@45113)

在 com.atlassian.jira.web.dispatcher.JiraWebworkActionDispatche 中会对 jspa 进行处理

service 函数会从请求中获取 Action 名称，后续使用/和. jspa 切割字符来获取 ActionName

```
67
68 public void service(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) throws ServletException, IOException {
69     Pair<HttpServletRequest, HttpServletResponse> wrap = this.wrap(httpServletRequest, httpServletResponse);
70     httpServletRequest = (HttpServletRequest)wrap.first();
71     httpServletResponse = (HttpServletResponse)wrap.second();
72     boolean doCleanup = httpServletRequest.getAttribute( @ "jira.webwork.cleanup") == null || httpServletRequest.getAttribute( @ "jira.webwork.cleanup").equals(Boolean.TRUE);
73     ServerMetricsDetailCollector serverMetricsDetailCollector = (ServerMetricsDetailCollector)ComponentAccessor.getComponent( @ ServerMetricsDetailCollector.class).orElseGet( @ noopServerMetricsDetailCollector::new);
74     GenericDispatcher gd = null;
75
76     try {
77         String actionName = this.getActionName(httpServletRequest);
78         serverMetricsDetailCollector.checkpointReached(RequestCheckpoints.beforeWorkActionPrepare@dispatcher.name());
79         gd = this.prepareDispatcher(httpServletRequest, httpServletResponse, actionName);
80         ActionResult ar = null;
81
82         try {
83             serverMetricsDetailCollector.checkpointReached(RequestCheckpoints.beforeWorkActionStarts.name());
84             gd.executeAction();
85             serverMetricsDetailCollector.checkpointReached(RequestCheckpoints.afterWorkActionExecute.name());
86             ar = gd.finish();
87         } catch (XsrfFailureException var17) {
88             httpServletRequest.getRequestDispatcher(XsrfErrorAction.FORWARD_PATH).forward(httpServletRequest, httpServletResponse);
89         } catch (WebSudoSessionException var18) {
90             ar = new ActionResult( @ null, "/secure/admin/WebSudoAuthenticate/default.jspa?webSudoDestination=" + this.getDestinationUrl(httpServletRequest), Collections.emptyList(), (Exception) null);
91         } catch (ActionNotFoundException var19) {
92             log.debug("Action '{}' was not found, returning 404", var19.getActionName());
93             this.sendErrorImpl(httpServletResponse, @ HttpStatus.SC_404, (String)null);
94         } catch (UnauthorizedActionException var20) {
95             httpServletRequest.getRequestDispatcher( @ "/login.jsp?permissionViolation=true&os_destination=" + this.getDestinationUrl(httpServletRequest)).forward(httpServletRequest, httpServletResponse);
96         } catch (Exception var21) {
97             this.onActionRecoverableThrowable(httpServletResponse, actionName, var21);
98         }
99     }
100 }
```

```
    }
    httpServletRequest.setAttribute( "jira.webwork.generic.dispatcher", gd);
}

private String getActionName(HttpServletRequest httpServletRequest) {
    String servletPath = (String)httpServletRequest.getAttribute( "javax.servlet.include.servlet_path");
    if (servletPath == null) {
        servletPath = httpServletRequest.getServletPath();
    }

    int beginIdx = servletPath.lastIndexOf("/");
    int endIdx = servletPath.lastIndexOf(".jsp");
    return servletPath.substring(beginIdx == -1 ? 0 : beginIdx + 1, endIdx == -1 ? servletPath.length() : endIdx);
}

private Pair<HttpServletRequest, HttpServletResponse> wrap(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse) {
    if (httpServletRequest instanceof MultiPartRequestWrapper) {
        return Pair.of(httpServletRequest, httpServletResponse);
    } else {
        httpServletResponse.setContentType(httpServletResponse.getContentType());
        String disableMultipartGetString = this.multipartDisableGetString();
        boolean disableMultipartGet = Boolean.valueOf(disableMultipartGetString);
        if (this.needsMultipartWrapper((HttpServletRequest)httpServletRequest, disableMultipartGet)) {
            try {
                httpServletRequest = new MultiPartRequestWrapper((HttpServletRequest)httpServletRequest, this.saveDir, this.getMaxSize());
            } catch (IOException var6) {
                ((HttpServletRequest)httpServletRequest).setAttribute( "webwork.action.ResultException", new ResultException("error", var6.getLocalizedMessage()));
            }
        }
    }
}
```

在 `webwork.dispatcher.GenericDispatcher#executeAction` 函数，会对 Action 进行检查

```
public void executeAction() throws Exception {
    if (!this.prepared) {
        throw new Exception("You must prepare the dispatcher first");
    } else {
        Action action = this.getActionFactory().getActionImpl(this.actionName);
        if (this.lazy) {
            this.lazyValueHolder = new GenericDispatcher.LazyValueHolder(action);
            ActionContext.getValueStack().pushValue(this.lazyValueHolder);
        } else {
            try {
                this.result = this.executeAction(action);
            } catch (Exception var7) {
                this.actionException = var7;
                return;
            }
        }
    }
}
```

系列 Action 工厂

```
package com.atlassian.jira.config.webwork;

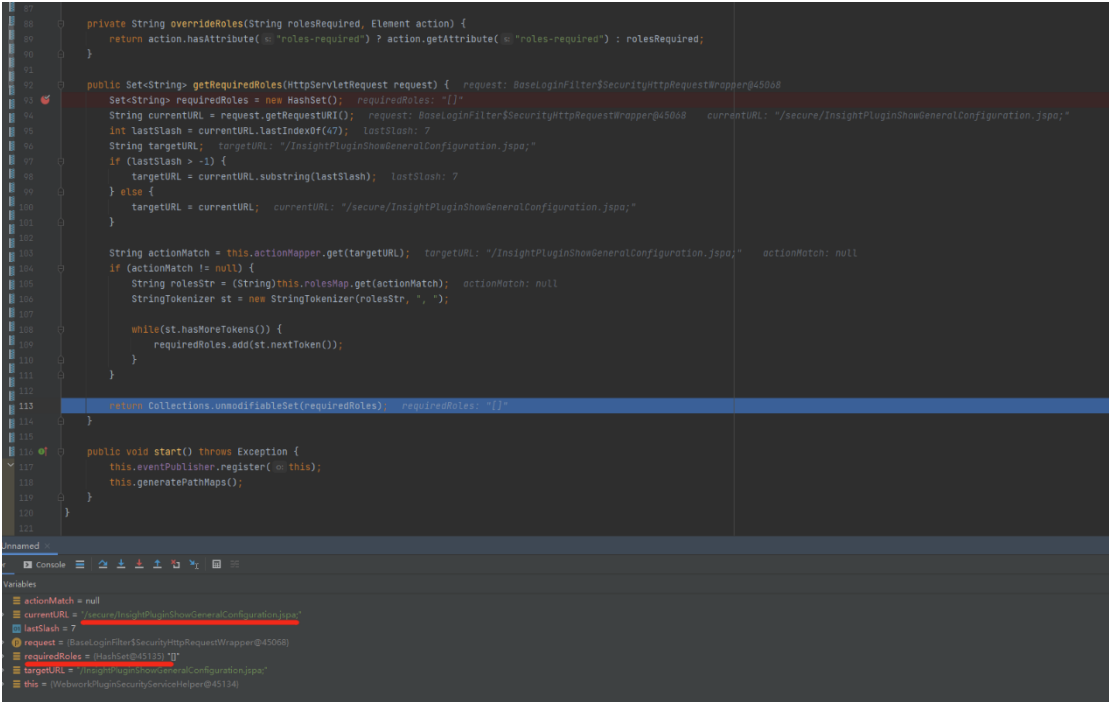
import ...

public class JiraActionFactory extends ActionFactory {
    private static final Logger log = LoggerFactory.getLogger(JiraActionFactory.class);
    private final ActionFactory factory;
    private final JiraActionFactory.JiraPluginActionFactory rootActionFactory;

    public JiraActionFactory() {
        this.rootActionFactory = new JiraActionFactory.JiraPluginActionFactory();
        ActionFactory factory = this.rootActionFactory;
        ActionFactory factory = new PrefixActionFactoryProxy(factory);
        ActionFactory factory = new JspActionFactoryProxy(factory);
        ActionFactory factory = new CommandActionFactoryProxy(factory);
        ActionFactory factory = new LookupAliasActionFactoryProxy(factory);
        factory = new CommandActionFactoryProxy(factory);
        ActionFactory factory = new ContextActionFactoryProxy(factory);
        ActionFactory factory = new PrepareActionFactoryProxy(factory);
        ActionFactory factory = new JiraActionFactory.SafeParameterSettingActionFactoryProxy(factory);
        Object factory = new ChainingActionFactoryProxy(factory);
    }
}
```

如上分析，其实目前已经知道在 Filter 中提取 URL 的方法是 `getRequestURL`，在 Servlet 中使用 `getServletPath`

在 URL 加入上“;”，传入 `/secure/InsightPluginShowGeneralConfiguration.jspa`; 那么在 Filter 中无法找到 `InsightPluginShowGeneralConfiguration.jspa`；对应的 Action，后续进入 Servlet 处理，而 `getServletPath` 会将;删除，这样也就绕过了 Filter 层的认证。

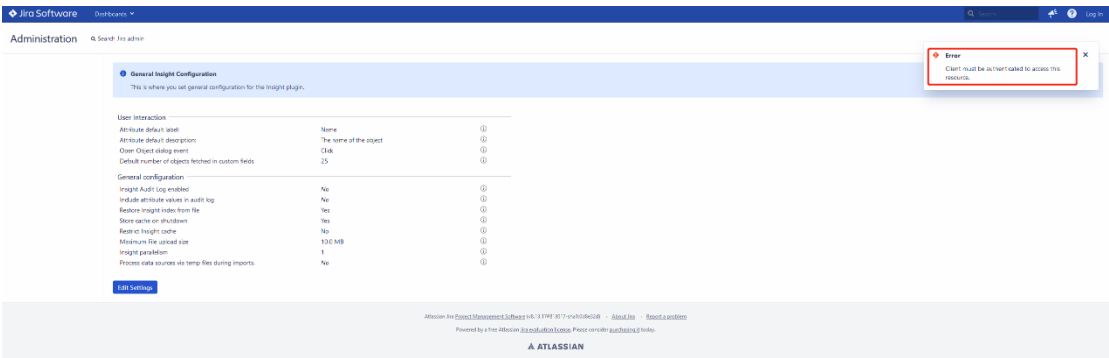


```
87 private String overrideRoles(String rolesRequired, Element action) {
88     return action.hasAttribute("@roles-required") ? action.getAttribute("@roles-required") : rolesRequired;
89 }
90
91 public Set<String> getRequiredRoles(HttpServletRequest request) {
92     request: BaseLoginFilter$SecurityHttpRequestWrapper@45068
93     Set<String> requiredRoles = new HashSet(); requiredRoles: "[]"
94     String currentURL = request.getRequestURI(); request: BaseLoginFilter$SecurityHttpRequestWrapper@45068 currentURL: "/secure/InsightPluginShowGeneralConfiguration.jspa;"
95     int lastSlash = currentURL.lastIndexOf(47); lastSlash: 7
96     String targetURL = currentURL.substring(0, lastSlash); targetURL: "/InsightPluginShowGeneralConfiguration.jspa;"
97     if (lastSlash > -1) {
98         targetURL = currentURL.substring(0, lastSlash); lastSlash: 7
99     } else {
100         targetURL = currentURL; currentURL: "/secure/InsightPluginShowGeneralConfiguration.jspa;"
101     }
102
103     String actionMatch = this.actionMapper.get(targetURL); targetURL: "/InsightPluginShowGeneralConfiguration.jspa;" actionMatch: null
104     if (actionMatch != null) {
105         String rolesStr = (String)this.rolesMap.get(actionMatch); actionMatch: null
106         StringTokenizer st = new StringTokenizer(rolesStr, ",");
107
108         while (st.hasMoreTokens()) {
109             requiredRoles.add(st.nextToken());
110         }
111     }
112
113     return Collections.unmodifiableSet(requiredRoles); requiredRoles: "[]"
114 }
115
116 public void start() throws Exception {
117     this.eventPublisher.register(this);
118     this.generatePathMaps();
119 }
120 }
121
122
```

Variables

- actionMatch = null
- currentURL = "/secure/InsightPluginShowGeneralConfiguration.jspa;"
- lastSlash = 7
- request = BaseLoginFilter\$SecurityHttpRequestWrapper@45068
- requiredRoles = HashSet@45133: "[]"
- targetURL = "/InsightPluginShowGeneralConfiguration.jspa;"
- this = WebworkPluginSecurityServiceHelper@45134

但实际效果上，这样还需要进行验证才能访问资源



继续动态调试发现

在 `LookupAliasActionFactoryProxy` 同样进行了权限检查

```
private ConcurrentMap<String, Entry> actionAliases = new ConcurrentHashMap();

public LookupAliasActionFactoryProxy(ActionFactory aFactory) { super(aFactory); }

public Action getActionImpl(String alias) throws Exception {
    Entry actionCommand = (Entry)this.actionAliases.get(alias);
    if (actionCommand == null) {
        actionCommand = this.getActionConfiguration().getActionCommand(alias);
        if (actionCommand == null) {
            throw new ActionNotFoundException(alias);
        }
    }

    Entry fromMap = (Entry)this.actionAliases.putIfAbsent(alias, actionCommand);
    if (fromMap != null) {
        actionCommand = fromMap;
    }
}

this.authorise(actionCommand);
return this.getNextFactory().getActionImpl(actionCommand.toActionFactoryString());
}

private void authorise(Entry actionCommand) {
    actionCommand: ActionConfiguration$Entry@45253
    Iterator var2 = actionCommand.getPermissionsRequired().iterator();
    actionCommand: ActionConfiguration$Entry@45253

    Integer permission;
    do {
        if (!var2.hasNext()) {
            return;
        }

        permission = (Integer)var2.next();
    } while(this.getPermissionManager().hasPermission(permission, this.getJiraAuthenticationContext().getUser()));

    throw new LookupAliasActionFactoryProxy.UnauthorisedActionException();
}
```

参考了其他大佬分析的文章，了解到在编写插件的时候可使用 `webwork1` 元素添加 `roles-required` 属性。这里直接使用受影响的插件 `insight8.9.10` 进行复现。

添加角色属性可参考 <https://developer.atlassian.com/server/jira/platform/webwork/>

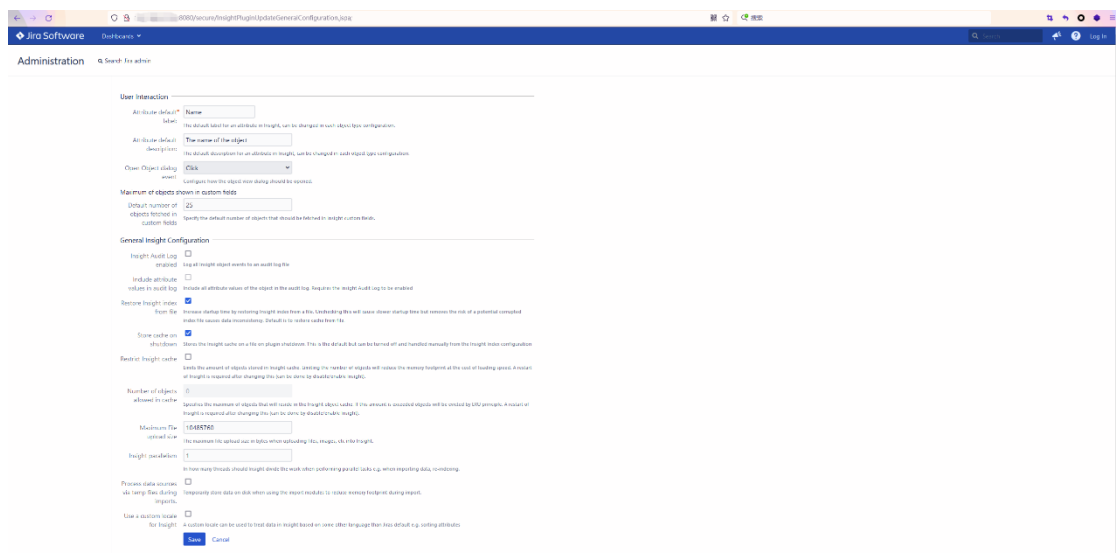
## 漏洞复现

在官网上下载插件 `Insight8.9.10` 版本

Insight - Asset Management				
Version history				
<button>Watch</button>				
> 8.10.7	Jira Data Center 8.12.0 - 9.1.0	2022-08-03	Minor bugfix release	
> 8.10.5	Jira Data Center 8.12.0 - 9.1.0	2022-07-25	Minor bugfix release	
> 8.10.1	Jira Data Center 8.12.0 - 8.22.6	2022-04-21	Minor bugfix release	
> 8.10.0	Jira Data Center 8.12.0 - 8.22.6	2022-03-21	Minor bugfix release	
> 8.10.7	Jira Server 8.12.0 - 9.1.0	2022-08-03	Minor bugfix release	
> 8.10.5	Jira Server 8.12.0 - 9.1.0	2022-07-25	Minor bugfix release	
> 8.10.1	Jira Server 8.12.0 - 8.22.6	2022-04-21	Minor bugfix release	
> 8.10.0	Jira Server 8.12.0 - 8.22.6	2022-03-21	Minor bugfix release	
> 8.9.10	Jira Data Center 8.12.0 - 8.22.6	2022-03-15	Minor bugfix release	
> 8.9.8	Jira Data Center 8.12.0 - 8.22.6	2022-02-17	Minor bugfix release	
> 8.9.5	Jira Data Center 8.12.0 - 8.22.6	2022-01-19	Minor bugfix release	



上传成功后，在未登陆的情况下，访问  
`/secure/InsightPluginUpdateGeneralConfiguration.jspa;`



成功绕过登陆认证限制。

## 修复建议

受影响用户可将产品更新至最新安全版本，具体参考官网公告

<https://jira.atlassian.com/browse/JRASERVER-73650>