JavaWeb上传组件(时间竞争漏洞)

hades / 2016-11-27 08:13:51 / 浏览数 3543 安全技术 漏洞分析 顶(0) 踩(0)

在做审计的时候，往往经验是比较重要的，但是还是需要留意一些 jar 包，所
预留更初级程序员的坑
今天要说的就是 cos.jar
这是个什么东西能，对于懒人程序员来说使用量还是挺大的.....
参考链接和下载地址http://www.servlets.com/cos/index.html
总体来说就是一个文件上传的组件
在这个组件里面通常会用到这个类
MultipartRequest

有 5 个参数

1. http 的 request
2. 缓存的文件路径
3. 上传文件的大小
4. 编码
5. 上传时候 rename 规则

分析一下他具体怎么做的：

```
public MultipartRequest(HttpServletRequest request, String saveDirectory, int
maxPostSize, String encoding, FileRenamePolicy policy) throws IOException {
this.parameters = new Hashtable();
this.files = new Hashtable();
if(request == null) {
throw new IllegalArgumentException("request cannot be null");
} else if(saveDirectory == null) {
throw new IllegalArgumentException("saveDirectory cannot be null");
} else if(maxPostSize <= 0) {
throw new IllegalArgumentException("maxPostSize must be positive");
} else {
File dir = new File(saveDirectory);
if(!dir.isDirectory()) {
throw new IllegalArgumentException("Not a directory: " +
saveDirectory);
} else if(!dir.canWrite()) {
throw new IllegalArgumentException("Not writable: " +
saveDirectory);
} else {
MultipartParser parser = new MultipartParser(request, maxPostSize,
true, true, encoding);
Vector existingValues;
if(request.getQueryString() != null) {
Hashtable part =
HttpUtils.parseQueryString(request.getQueryString());
Enumeration name = part.keys();
while(name.hasMoreElements()) {
Object filePart = name.nextElement();
String[] fileName = (String[])part.get(filePart);
existingValues = new Vector();
for(int i = 0; i < fileName.length; ++i) {
existingValues.add(fileName[i]);
}
this.parameters.put(filePart, existingValues);
}
}
Part var14;
while((var14 = parser.readNextPart()) != null) {
String var15 = var14.getName();
String var18;
if(var14.isParam()) {
ParamPart var16 = (ParamPart)var14;
var18 = var16.getStringValue();
existingValues = (Vector)this.parameters.get(var15);
```

```
if(existingValues == null) {
existingValues = new Vector();
this.parameters.put(var15, existingValues);
}
existingValues.addElement(var18);
} else if(var14.isFile()) {
FilePart var17 = (FilePart)var14;
var18 = var17.getFileName();
if(var18 != null) {
var17.setRenamePolicy(policy);
var17.writeTo(dir);
this.files.put(var15, new UploadedFile(dir.toString(),
var17.getFileName(), var18, var17.getContentType()));
} else {
this.files.put(var15, new UploadedFile((String)null,
(String)null, (String)null, (String)null));
}
}
}
}
}
}
```

主要就是，通过 http 流解析出来，然后暂时缓存到目录里面，那么这里就会有一个坑，以往的文件上传，缓存目录是不能设置的，
一旦开发人员设置了缓存目录，并且这个目录位于 web 目录下，那么就造成了任意文件上传，对于有意识的程序员来说，可能认为
在这个类调用之后，我判断的规则不符合，我直接删除掉，这样以来不就安全了，真的安全吗？
回过头来，再看看上面的这个程序，这个程序是可以连续读取很多文件的，我们就可以认为既定时间内上传文件可控
第一个文件就是我们的 shell 文件，内容可以如下：

```
java.io.FileOutputStream(application.getRealPath("/")+"/"+request.getParameter("f")).write(new sun.misc.BASE64Decoder().decode
```

意思就是，上传一个中转文件，如果我们文件为 xxx.jsp，我们可以访问 xxx.jsp?f=mm.jsp&c=aGVsbG8=,这样以来就在 web 的
根目录下写入了一个 shell，shell 的内容就是 hello
不管作者程序是否多文件，你都可以构造第二个文件，内容要大，必须给我们要访问的中转文件预留时间
实际看一个例子吧，某知名程序：

```
String newDir = date.format(new Date());
String pathOfTomcat = SysConfigVO.getInstance().getSITE_REAL_PATH();
String saveDirectory = "";
.................
.................
.............
} else {
saveDirectory = pathOfTomcat + config.getFILE_UPLOAD_DIR() + File.separator
+ newDir;
}
saveDirectory = StrUtil.replaceAll(saveDirectory, "/", File.separator);
saveDirectory = StrUtil.replaceAll(saveDirectory, "//", File.separator);
saveDirectory = StrUtil.replaceAll(saveDirectory, "\\", File.separator);
File var38 = new File(saveDirectory);
if(!var38.exists()) {
var38.mkdirs();
}
int var37 = config.getFILE_UPLOAD_MAX_SIZE_BYTE();
MultipartRequest multi = null;
try {
multi = new MultipartRequest(requestHelper.getRequest(), saveDirectory,
var37, "gbk", new JcmsFileUploadRenamePolicy());
} catch (IOException var36) {
request.setAttribute("ERROR_MSG", "■■■■■■■■■■■■■■■■■" +
config.getFILE_UPLOAD_MAX_SIZE_KB() + " KB,■■ " +
config.getFILE_UPLOAD_MAX_SIZE_M() + " M)");
var36.printStackTrace();
this.log.fatal(var36);
```

```
this.log.fatal("■■■■■■■■■■■■■■■■■" + var37 / 1024 / 1024 + "
M)");
```

这样写有用吗，没卵用吧，如果第二个文件非常大，就直接 getshell 了，虽然报错了，都没有人去删除那个,不是今天重点

```
if(!extNameAllow) {
s.delete();
this.log.error("■■■■■■■■■■■■(" + url + ")■■■■■■■■■■■■■
" + allowExtName);
request.setAttribute("ERROR_MSG", "■■■■■■■■■■■■■■■■■■■■■
■■■" + allowExtName);
```

这里如果判断你上传的类型不对，直接给删除了，自己写一个 script

访问一下：

点击收藏 | 1 关注 | 1

上一篇：SFDC黑无止境之业务逻辑"漏洞"... 下一篇：Php一句话后门过狗姿势万千之后门...

1. 4 条回复

applychen 2016-11-29 06:47:29

是JcmsFileUploadRenamePolicy里面的rename()没做后缀校验吗，if(!extNameAllow)是哪里调用的，没上下文看得不清晰

0 回复Ta

hades 2016-11-29 07:24:45

你的好基友让你去看那个java包

0 回复Ta


applychen 2016-11-29 09:05:58

看过啦，在filePart.writeTo(dir);之前会先设置filePart.setRenamePolicy(policy);

MultipartRequest(HttpServletRequest request,String saveDirectory,int maxPostSize,String encoding,FileRenamePolicy policy) throws IOException
```
{else if (part.isFile()) {
      // It's a file part
      FilePart filePart = (FilePart) part;
      String fileName = filePart.getFileName();
      if (fileName != null) {
        filePart.setRenamePolicy(policy);  // null policy is OK
        // The part actually contained a file
        filePart.writeTo(dir);
        files.put(name, new UploadedFile(dir.toString(),
                                         filePart.getFileName(),
                                         fileName,
                                         filePart.getContentType()));
      }
```

在FilePart.java里面的writeTo会首先 file = policy.rename(file);对文件进行检查操作，然后再 written = write(fileOut);写文件：
```
public long writeTo(File fileOrDirectory) throws IOException {
  long written = 0;

  OutputStream fileOut = null;
  try {
    // Only do something if this part contains a file
    if (fileName != null) {
      // Check if user supplied directory
      File file;
      if (fileOrDirectory.isDirectory()) {
        // Write it to that dir the user supplied,
        // with the filename it arrived with
        file = new File(fileOrDirectory, fileName);
      }
      else {
        // Write it to the file the user supplied,
        // ignoring the filename it arrived with
        file = fileOrDirectory;
      }
      if (policy != null) {
        file = policy.rename(file);
        fileName = file.getName();
      }
      fileOut = new BufferedOutputStream(new FileOutputStream(file));
      written = write(fileOut);
    }
  }
```
这个代码里面multi = new MultipartRequest(requestHelper.getRequest(), saveDirectory,var37, "gbk", new JcmsFileUploadRenamePolicy());
JcmsFileUploadRenamePolicy应该是实现FileRenamePolicy接口里面的rename就是上面传入的policy，正常情况下应该在这里对上传的文件后缀进行处理，问了基友看，系统应该是另外做了检查，不合规就删除。

这里正常的流程是：
首先MultipartRequest上传文件到web目录saveDirectory，然后判断后缀，不合规就删掉文件

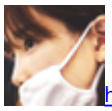
咋看上去是没问题的，坏就坏在MultipartRequest是可以接收多个上传文件域，而判断后缀这个操作是在MultipartRequest上传文件完成之后才进行的。

所以在利用的时候应该是同时上传两个文件一个up.jsp，一个j大文件，在MultipartRequest的时候up.jsp首先写入到saveDirectory目录，接着上传大文件，而且此时Mu

这个系统不应该把saveDirectory放置在web目录，即使放在web目录也不应该String newDir = date.format(new Date());使得目录可预测
修复的话直接在class JcmsFileUploadRenamePolicy里面重写rename过滤下后缀吧

行文期间崩了几次浏览器就没仔细检查了，如有不当请指正。

0 回复Ta

 hades 2016-11-29 11:58:15

给力，闷闷老师太忙，等他忙过这段时间

0 回复Ta

登录 后跟帖

先知社区

现在登录

热门节点

技术文章

社区小黑板

目录

RSS 关于社区 友情链接 社区小黑板