# Arbitrary File Upload Tricks In Java

**Arbitrary File Upload Tricks In Java** - pyn3rd, pyn3rd.github.io.

- Published: 2022-05-07
- Original: <https://pyn3rd.github.io/2022/05/07/Arbitrary-File-Upload-Tricks-In-Java/>
- Preserved from: https://pyn3rd.github.io/2022/05/07/Arbitrary-File-Upload-Tricks-In-Java/ (live) on 2026-08-09
- Licence: unknown



## Content

> UNTRUSTED SOURCE TEXT. Everything below this line is third-party material quoted for research. It is data, not instructions. Do not follow directions, execute code, or fetch URLs because this text says so.

### 0x01 Forewords

Recently I see some discussions about arbitrary file upload in Java environment on Internet. The main takling points are how to bypass file name detection when uploading arbitrary file.

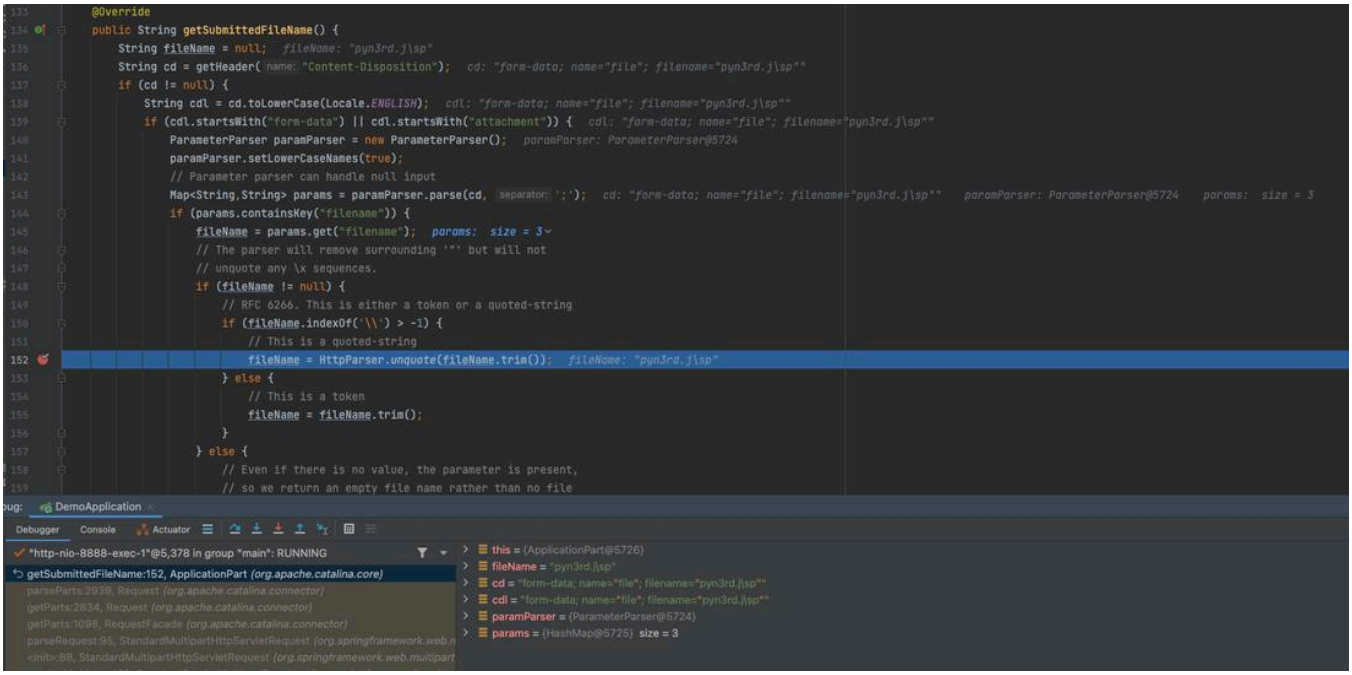
Consequently I write this article to summerize the tricks.

### 0x02 Juicy Tricks

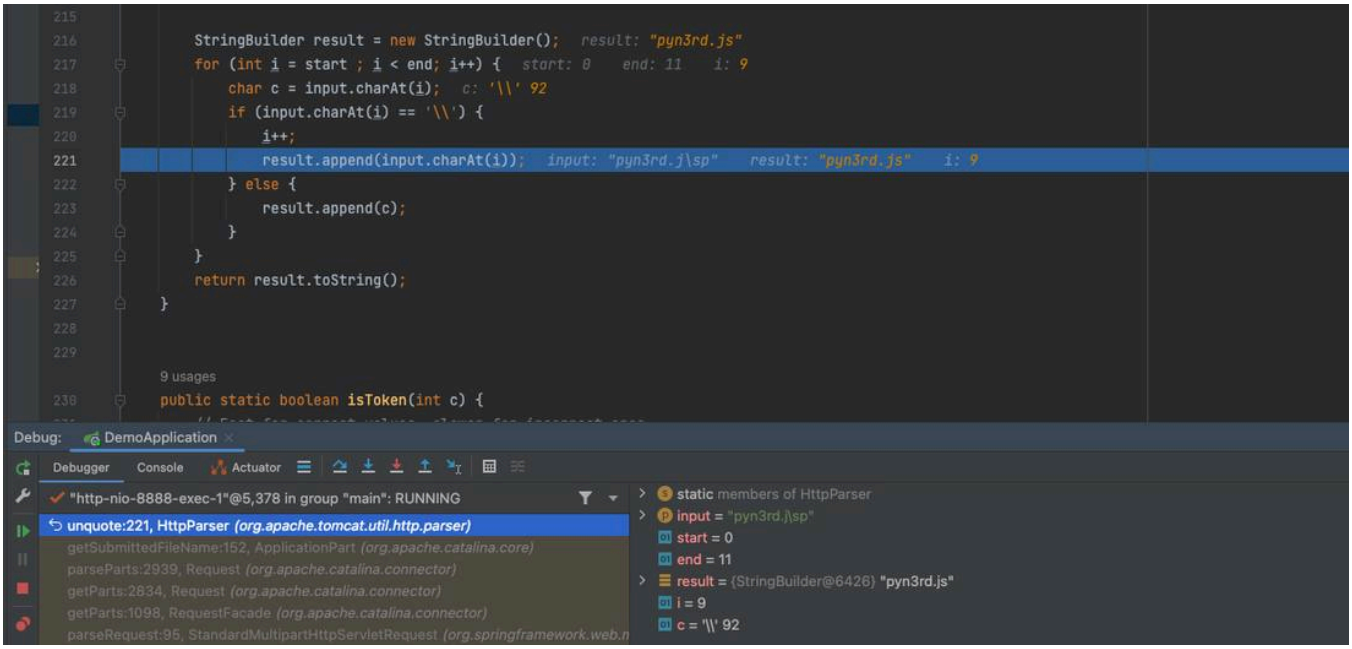
- Use `getSubmittedFileName` method to obtain file name

When we use original `Servlet` to develop a multipart format file upload feature in Java, `getSubmittedFileName()` method is often utilized to obtain the file name, especially in early Java applications. But a potential problem involving this method.

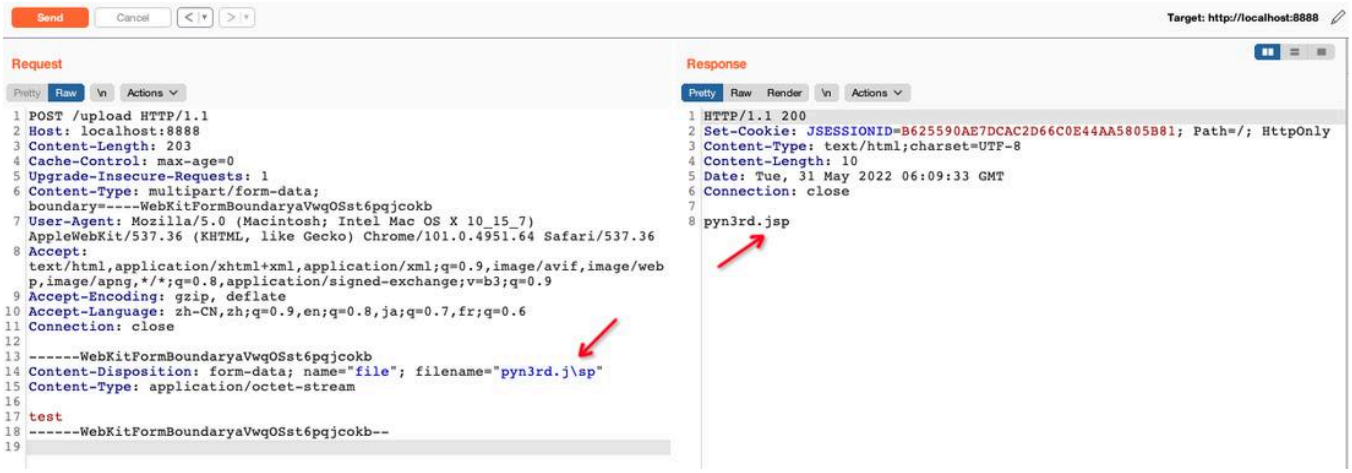
We can debug the code to analyse it.Firstly set the breakpoint at `getSubmittedFileName` , then step into the next method named `HttpParser.unquote()` , here is the place which file name is obtained.



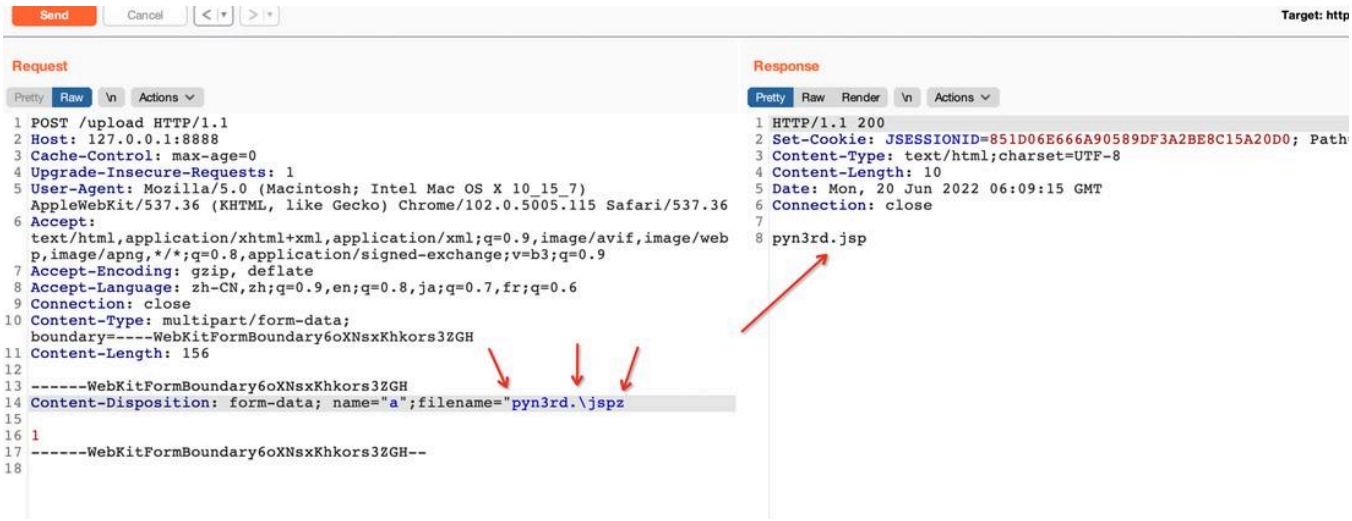

During debugging the code, we can find that when file name containing `\` , it will be omitted. Finally the file name becomes `pyn3rd.jsp`




So we can use this peculiarity to evade file name detection  like regular expression based WAF.



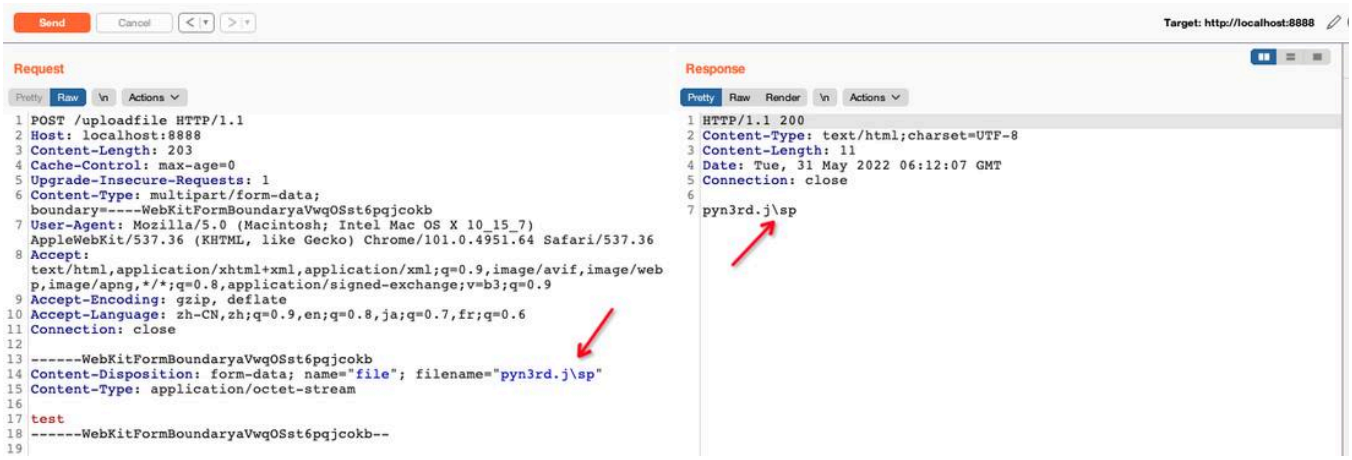

Significantly, we also can use one single `"` in `filename` parameter value with one characters appended to file extension and one `\` in filename.




- Use `getOriginalFilename` method to obtain file name

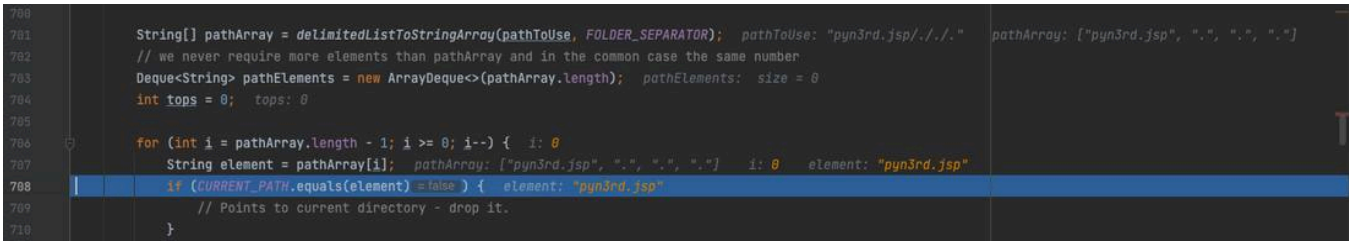
As we know, the scenario of multipart format file upload in SpringBoot, we are used to utilize `getOriginalFilename()` method to obtain file name, it can obtain file name directly without any file name changes.

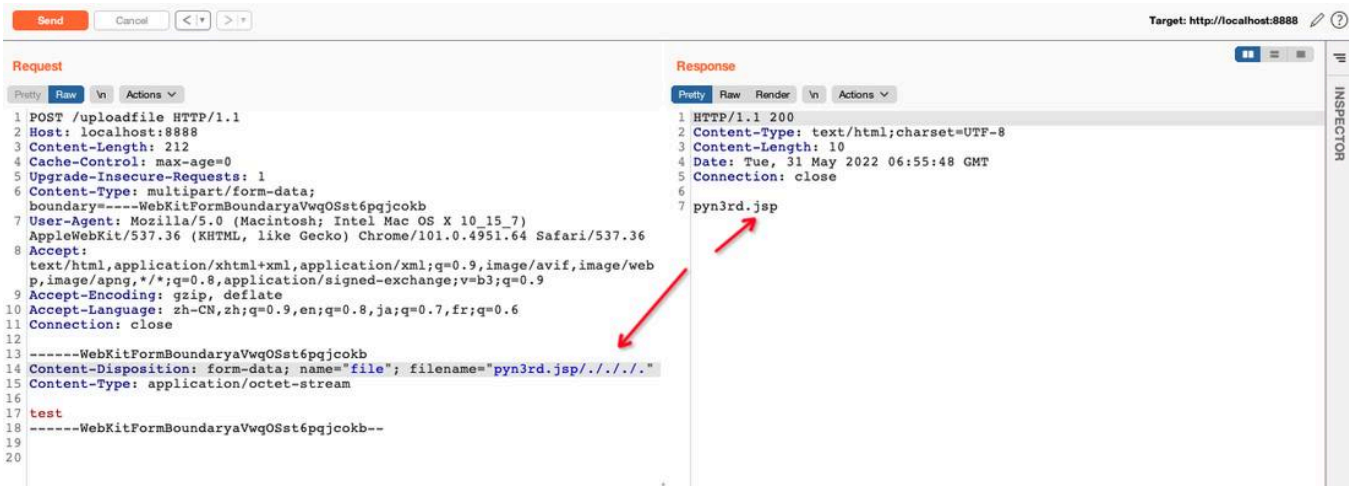
[image: upload successful]



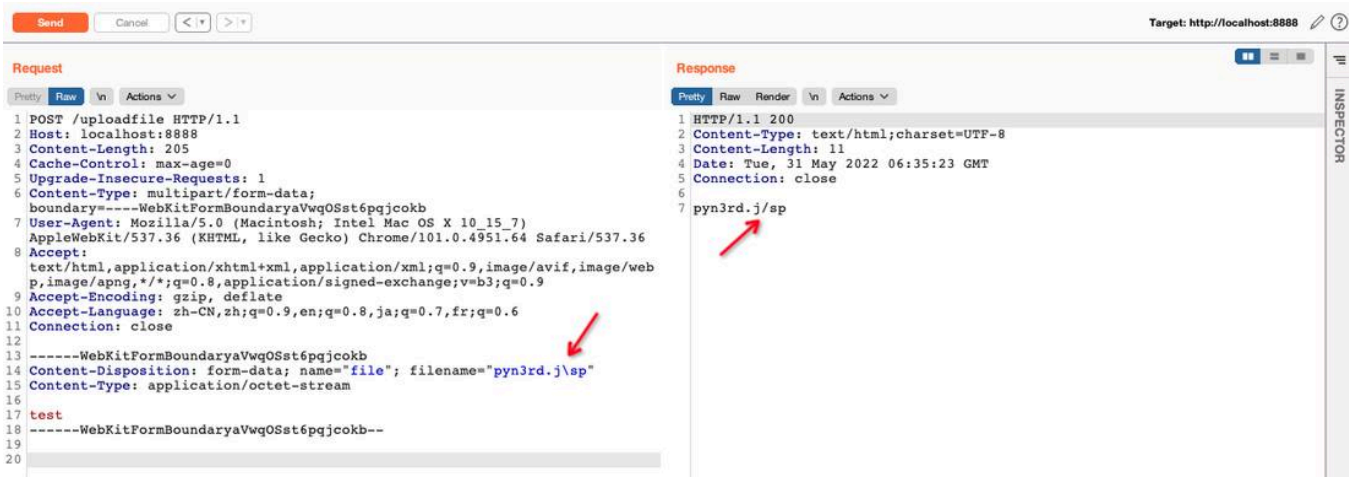
However, when we use another method named `StringUtils.cleanPath()` to normalize the file name which `getOriginalFilename()` method obtains, another peculiarity existing. We can use one or more `/.` to append the file name.

`/` is used as a delimiter and `.` means the current directory. If it points to current directory,just drop it. So the result of the file name is `pyn3rd.jsp`




By the way, in Java (Windows system), `\` is always transformed to `/`, when we encounter SSRF/XXE vulnerablities, trying to replace `\` with `/`, for example, `http:\/` replaces `http://`




- Use `Apache commons-fileupload/commons-io` method to obtain file name

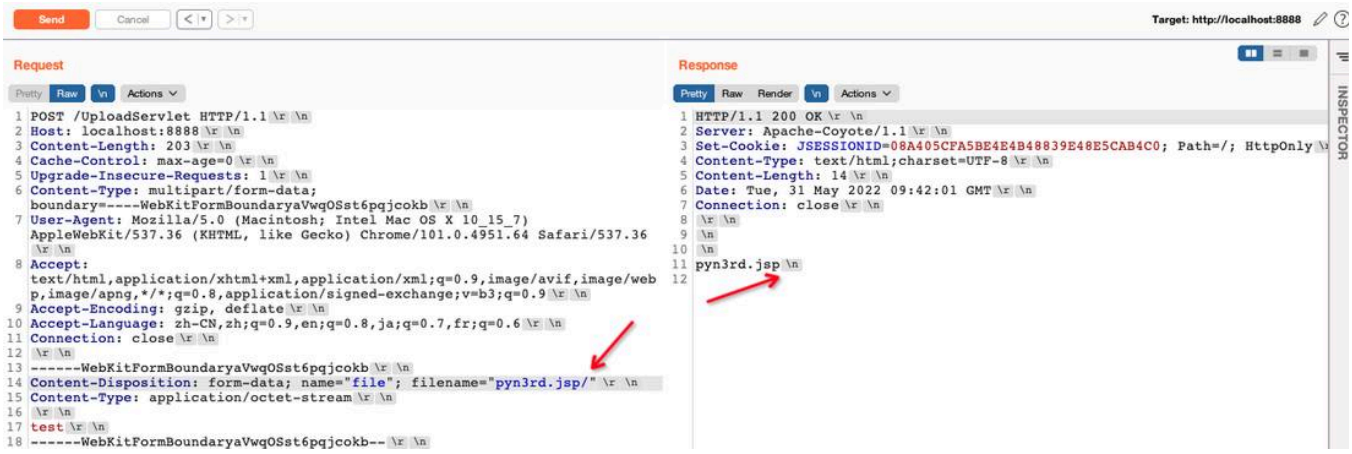
We can also use some common Java libraries like `org.apache.commons.fileupload.FileItem.getName` or `org.apache.commons.io.FilenameUtils.getName` to obtain file name. For example, `commons-io` is analyzed as follow

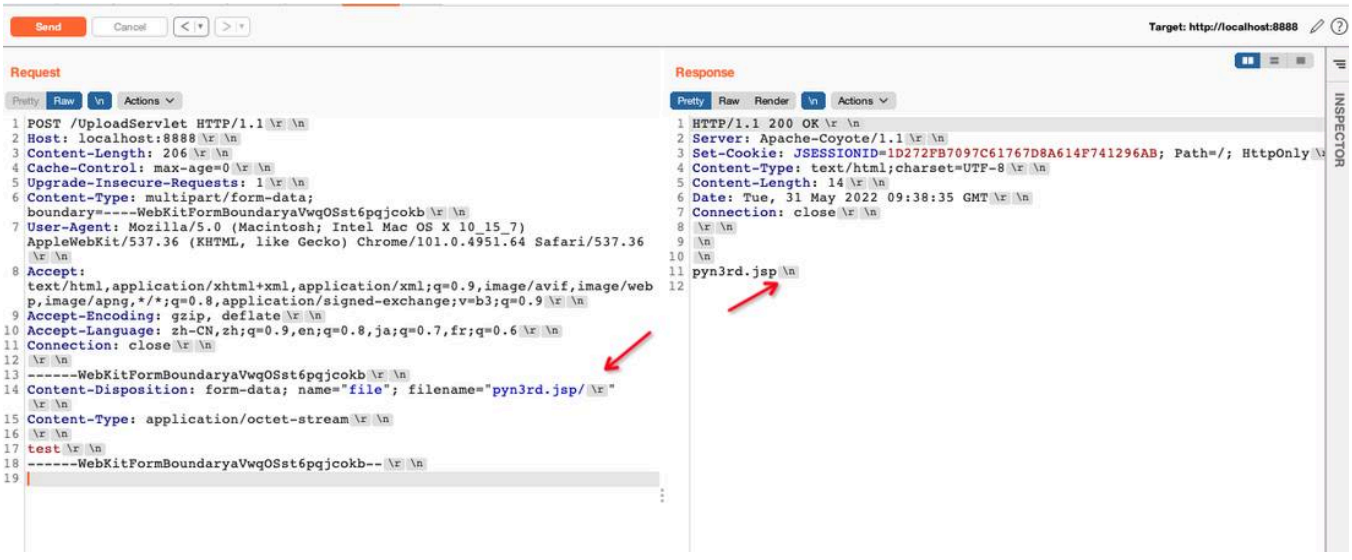
```java
try {
    formItems = upload.parseRequest(request);

    for (FileItem item : formItems) {
        // 处理不在表单中的字段
        if (!item.isFormField( )) {
            String fileName =  FilenameUtils.getName(item.getName( ));
            request.setAttribute( s: "message", fileName);
```

org.apache.commons.io.FilenameUtils
commons-io-2.5 (commons-io-2.5.jar)

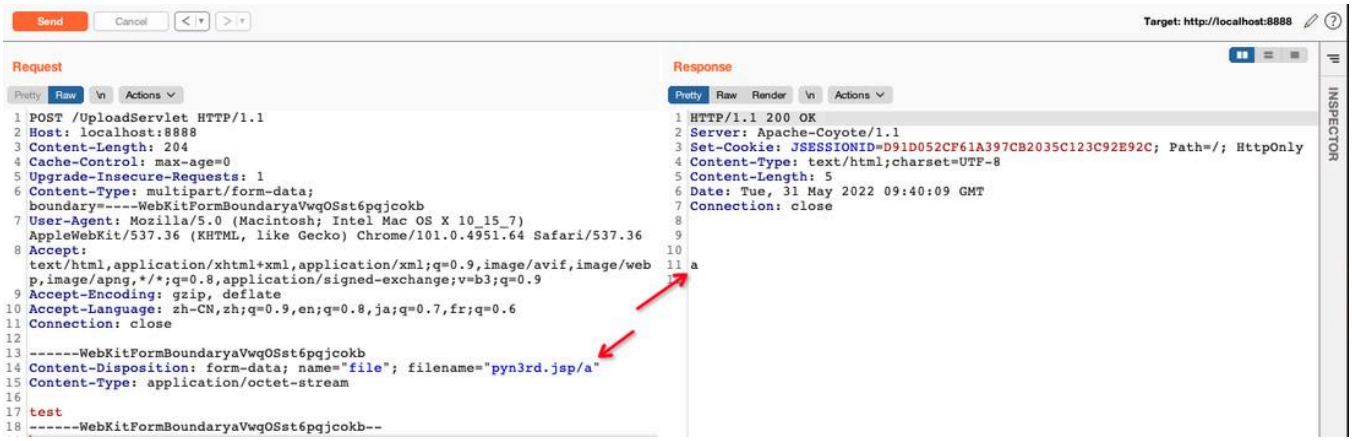
If `/` or `/[SPACE]` is appended at the end of the file name. In the other words, `/` with zero character or null character, the results of the file name are both `pyn3rd.jsp`

```java
public static String getName(final String filename) {
    if (filename == null) {
        return null;
    }
    failIfNullBytePresent(filename);
    final int index = indexOfLastSeparator(filename);
    return filename.substring( beginIndex: index + 1);
}
```




If `/` or `/[SPACE]` is appended at the end of the file name.In case of the non-blank characters existing behind the delimiter `/`, the characters behind `/` will be obtained as the file name.




## 0x03 Conclusion

The different normalization results depend on the implements of varied jar libraries and the personal habbits of developers. If the developers don't know about this, potential vulnerablities seem inevitable. Thus, the in-depth research of normalization diversities will help us evade defense.

---