目录

Java Zip Slip漏洞案例分析及实战挖掘

TGAO (/u/10667) / 2023-01-29 20:32:00 / 发表于江苏 / 浏览数 6131

前言

Zip Slip的漏洞成因非常简单，这个漏洞绑定的业务功能点：上传压缩包文件，后端解压压缩包保存其中的文件到服务器本地。

漏洞成因：待上传的压缩包中可以构造条目名，后端保存文件的时候，常常将条目名提取出来并和保存目录拼接作为最后的保存文件路径，但是压缩包是可控的，从而其中保存的原始条目名也是可控的，因此可以在文件名处利用 `../` 跳转到任意目录，从而向任意目录写入新文件或者覆盖旧文件。具体案例可见下文。

在Zip Slip公布者文章 (https://security.snyk.io/research/zip-slip-vulnerability)中，提到，Java中的Zip Slip漏洞尤其普遍：

> The vulnerability has been found in multiple ecosystems, including JavaScript, Ruby, .NET and Go, but is especially prevalent in Java, where there is no central library offering high level processing of archive (e.g. zip) files. The lack of such a library led to vulnerable code snippets being hand crafted and shared among developer communities such as StackOverflow.

本文从原生的Java.util.zip->zt-zip->spring integration zip进行Zip Slip漏洞分析，并在最后附上此漏洞的代审案例。

生成恶意zip

```
import zipfile

if __name__ == "__main__":
    try:
        zipFile = zipfile.ZipFile("poc.zip", "a", zipfile.ZIP_DEFLATED)  ##生成的zip文件
        info = zipfile.ZipInfo("poc.zip")
        zipFile.write("D:/tgao/pass/1", "../password", zipfile.ZIP_DEFLATED)  ##压缩的文件和在zip中显示的文件名
        zipFile.close()
    except IOError as e:
        raise e
```

上述生成的恶意zip，在Zip Slip中，会取出 `../password` ，并与保存目录拼接，其中获取 `../password` 的java方法类似与 `zipEntry.getName()` 。

原生的Java.util.zip

漏洞代码：实际场景下的的zip包是可控的，如通过文件上传等功能

```
package zip;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Enumeration;
import java.util.zip.ZipEntry;
import java.util.zip.ZipFile;

public class Zip1 {
    public static void main(String[] args) throws IOException {
        //解压zip的包
        String fileAddress = "D:/pythonProject/exp/ctf/poc.zip";
        //zip文件解压路径
        String unZipAddress = "D:/tgao/pass/";
        //去目录下寻找文件
        File file = new File(fileAddress);
        ZipFile zipFile = null;
        try {
            zipFile = new ZipFile(file);//设置编码格式
        } catch (IOException exception) {
            exception.printStackTrace();
            System.out.println("解压文件不存在!");
        }
        Enumeration e = zipFile.entries();
        while(e.hasMoreElements()) {
            ZipEntry zipEntry = (ZipEntry)e.nextElement();
            File f = new File(unZipAddress + zipEntry.getName());
            f.getParentFile().mkdirs();
            f.createNewFile();
            InputStream is = zipFile.getInputStream(zipEntry);
            FileOutputStream fos = new FileOutputStream(f);
            int length = 0;
            byte[] b = new byte[1024];
            while((length=is.read(b, 0, 1024))!=-1) {
                fos.write(b, 0, length);
            }
            is.close();
            fos.close();
        }
        if (zipFile != null) {
            zipFile.close();
        }
    }
}
```

漏洞成因： `File f = new File(unZipAddress + zipEntry.getName());` 中 `zipEntry.getName()` 的值是可控的，从而造成路径穿越，最终写入任意文件。

zt-zip

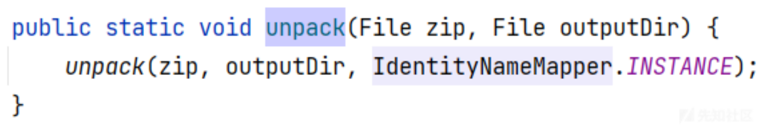
引入依赖：

```
<dependency>
  <groupId>org.zeroturnaround</groupId>
  <artifactId>zt-zip</artifactId>
  <version>1.12</version>xml
</dependency>
```

zt-zip组件中的解压功能，是在原生的java.util.zip基础上进行的封装。
漏洞代码：实际场景下的的zip包是可控的，如通过文件上传等功能。

```
package zip;

import org.zeroturnaround.zip.ZipUtil;

import java.io.File;

public class Zip2 {
    public static void main(String[] args) {
        File zip = new File("D:/pythonProject/exp/ctf/poc.zip");
        File dir = new File("D:/tgao/pass");
        ZipUtil.unpack(zip, dir);
    }
}
```

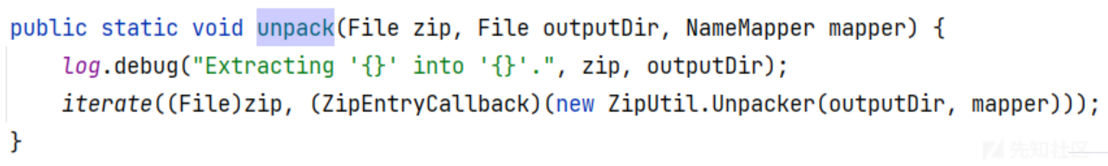
跟进 `org.zeroturnaround.zip.ZipUtil#unpack(java.io.File, java.io.File)`



```
public static void unpack(File zip, File outputDir) {
    unpack(zip, outputDir, IdentityNameMapper.INSTANCE);
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129110441-aca86b52-9f81-1.png)

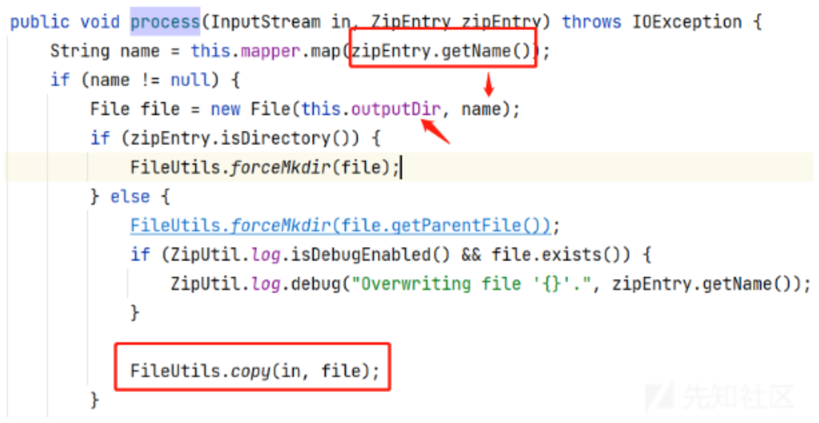
继续跟进 `org.zeroturnaround.zip.ZipUtil#unpack(java.io.File, java.io.File, org.zeroturnaround.zip.NameMapper)`



```
public static void unpack(File zip, File outputDir, NameMapper mapper) {
    log.debug("Extracting '{}' into '{}'.", zip, outputDir);
    iterate((File)zip, (ZipEntryCallback)(new ZipUtil.Unpacker(outputDir, mapper)));
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129110738-16132fe6-9f82-1.png)

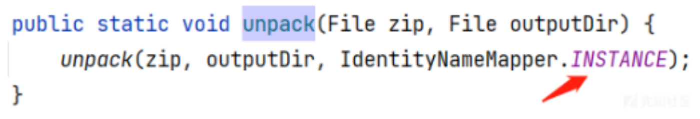
在上述方法中使用 `new ZipUtil.Unpacker(outputDir, mapper)` 创建 `ZipEntryCallback` 对象 (`ZipUtil.Unpacker`)
可以先看其中的 `org.zeroturnaround.zip.ZipUtil.Unpacker#process` 方法



```
public void process(InputStream in, ZipEntry zipEntry) throws IOException {
    String name = this.mapper.map(zipEntry.getName());
    if (name != null) {
        File file = new File(this.outputDir, name);
        if (zipEntry.isDirectory()) {
            FileUtils.forceMkdir(file);
        } else {
            FileUtils.forceMkdir(file.getParentFile());
            if (ZipUtil.log.isDebugEnabled() && file.exists()) {
                ZipUtil.log.debug("Overwriting file '{}'.", zipEntry.getName());
            }

            FileUtils.copy(in, file);
        }
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129111047-86d5afa6-9f82-1.png)

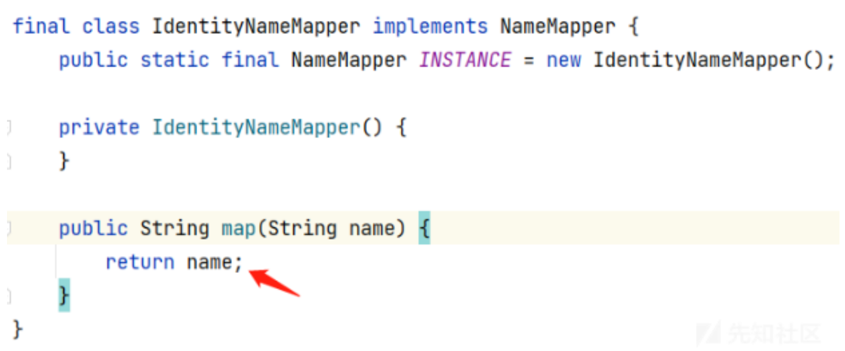
上述代码中的 `this.mapper` 在调用 `org.zeroturnaround.zip.ZipUtil#unpack(java.io.File, java.io.File)` 方法中传入的



```
public static void unpack(File zip, File outputDir) {
    unpack(zip, outputDir, IdentityNameMapper.INSTANCE);
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129110947-632debc2-9f82-1.png)

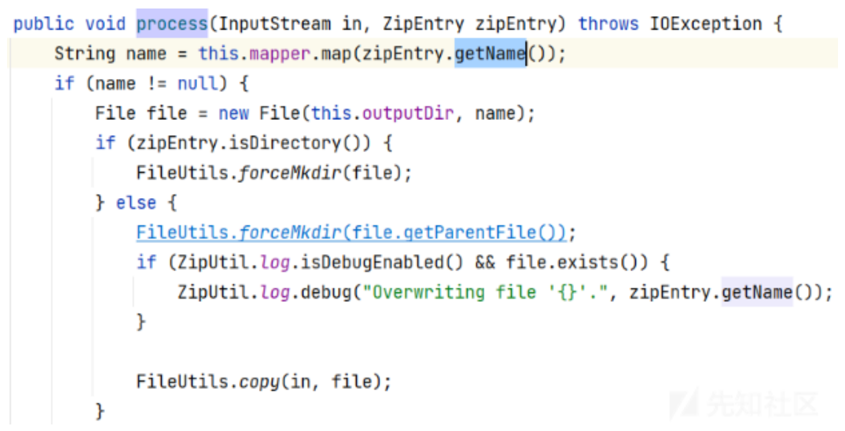
进入 `org.zeroturnaround.zip.IdentityNameMapper`



```java
final class IdentityNameMapper implements NameMapper {
    public static final NameMapper INSTANCE = new IdentityNameMapper();

    private IdentityNameMapper() {
    }

    public String map(String name) {
        return name;
    }
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129111145-a92fcdb6-9f82-1.png)

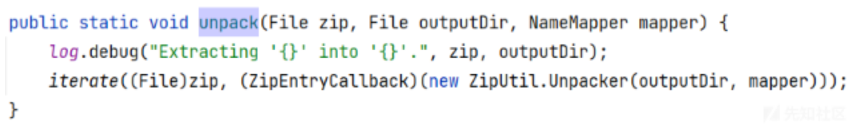
上述的map方法直接将传入的name参数返回并没有任何的过滤。
因此，再看 `org.zeroturnaround.zip.ZipUtil.Unpacker#process` 方法对 `zipEntry.getName()` 没有任何的过滤。所以导致了Zip Slip漏洞的产生。



```java
public void process(InputStream in, ZipEntry zipEntry) throws IOException {
    String name = this.mapper.map(zipEntry.getName());
    if (name != null) {
        File file = new File(this.outputDir, name);
        if (zipEntry.isDirectory()) {
            FileUtils.forceMkdir(file);
        } else {
            FileUtils.forceMkdir(file.getParentFile());
            if (ZipUtil.log.isDebugEnabled() && file.exists()) {
                ZipUtil.log.debug("Overwriting file '{}'.", zipEntry.getName());
            }

            FileUtils.copy(in, file);
        }
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129111221-bed5d1a6-9f82-1.png)

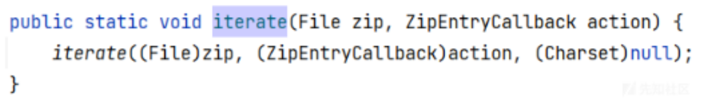
再回来看看 `org.zeroturnaround.zip.ZipUtil#unpack(java.io.File, java.io.File, org.zeroturnaround.zip.NameMapper)`



```java
public static void unpack(File zip, File outputDir, NameMapper mapper) {
    log.debug("Extracting '{}' into '{}'.", zip, outputDir);
    iterate((File)zip, (ZipEntryCallback)(new ZipUtil.Unpacker(outputDir, mapper)));
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129111327-e66d9898-9f82-1.png)

跟进 `org.zeroturnaround.zip.ZipUtil#iterate(java.io.File, org.zeroturnaround.zip.ZipEntryCallback)`

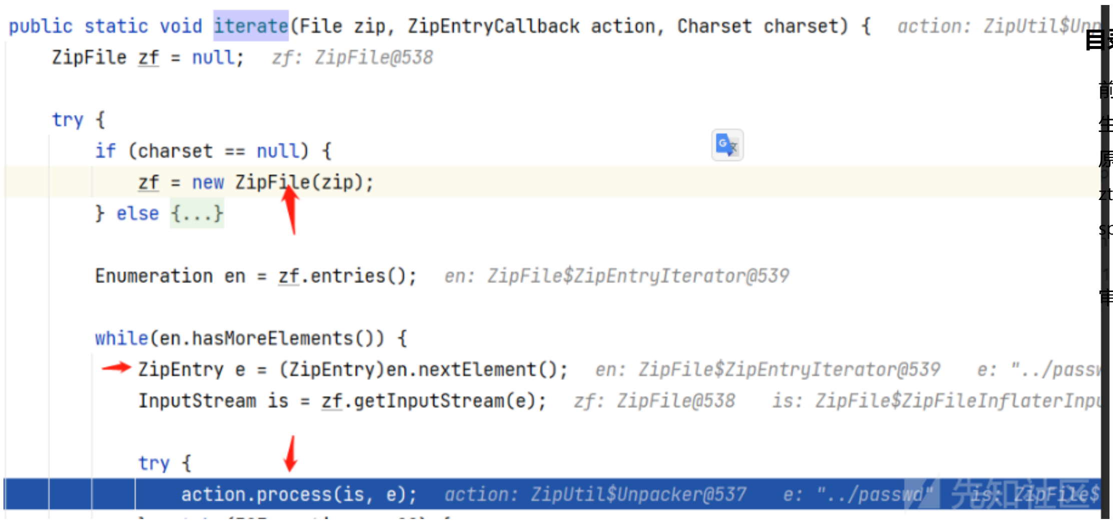


```java
public static void iterate(File zip, ZipEntryCallback action) {
    iterate((File)zip, (ZipEntryCallback)action, (Charset)null);
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129111354-f66e2d70-9f82-1.png)

继续跟进 `org.zeroturnaround.zip.ZipUtil#iterate(java.io.File, org.zeroturnaround.zip.ZipEntryCallback, java.nio.charset.Charset)`

目录

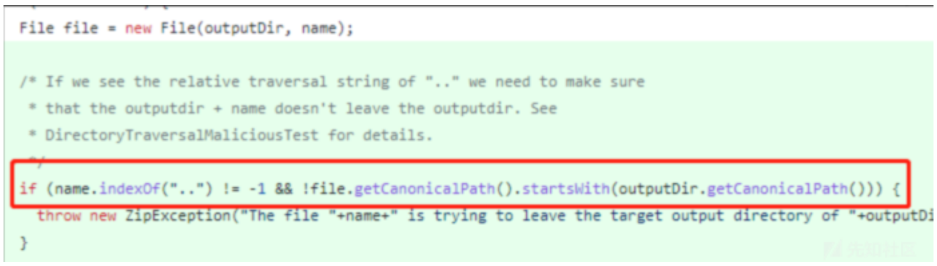


```
public static void iterate(File zip, ZipEntryCallback action, Charset charset) {   action: ZipUtil$Unp
    ZipFile zf = null;   zf: ZipFile@538

    try {
        if (charset == null) {
            zf = new ZipFile(zip);
        } else {...}

        Enumeration en = zf.entries();   en: ZipFile$ZipEntryIterator@539

        while(en.hasMoreElements()) {
            ZipEntry e = (ZipEntry)en.nextElement();   en: ZipFile$ZipEntryIterator@539    e: "../passw
            InputStream is = zf.getInputStream(e);   zf: ZipFile@538    is: ZipFile$ZipFileInflaterInpu

            try {
                action.process(is, e);   action: ZipUtil$Unpacker@537    e: "../passwd"   is: ZipFile$
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129111423-07a225ce-9f83-1.png)

可以看到调用了原生的 `java.util.zip.ZipFile#ZipFile(java.io.File)` 等API
此方法中也没有任何的过滤，直接将zip流内容和ZipEntry传入了 `org.zeroturnaround.zip.ZipUtil.Unpacker#process` (`Unpacker#process` 在上文已讲过)。

在zt-zip在1.13版本中进行了修复：https://github.com/zeroturnaround/zt-zip/commit/759b72f33bc8f4d69f84f09fcb7f010ad45d6fff# (https://github.com/zeroturnaround/zt-zip/commit/759b72f33bc8f4d69f84f09fcb7f010ad45d6fff#)



```
File file = new File(outputDir, name);

/* If we see the relative traversal string of ".." we need to make sure
 * that the outputdir + name doesn't leave the outputdir. See
 * DirectoryTraversalMaliciousTest for details.
 */
if (name.indexOf("..") != -1 && !file.getCanonicalPath().startsWith(outputDir.getCanonicalPath())) {
  throw new ZipException("The file "+name+" is trying to leave the target output directory of "+outputDi
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129111523-2b2c5f50-9f83-1.png)

spring integration zip

CVE-2018-1261

引入依赖

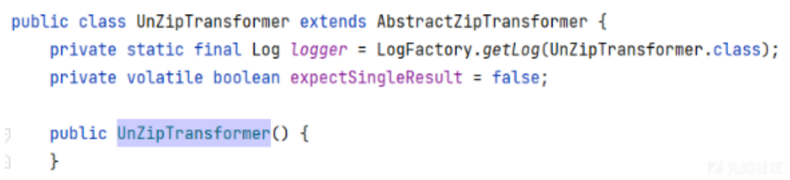
```
<dependency>
   <groupId>org.springframework.integration</groupId>
   <artifactId>spring-integration-zip</artifactId>
   <version>1.0.0.RELEASE</version>
</dependency>

<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-api</artifactId>
   <version>1.7.30</version>
</dependency>
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-simple</artifactId>
   <version>1.7.30</version>
   <type>jar</type>
</dependency>
```

`spring-integration-zip` 依赖于 `zt-zip`

漏洞代码：实际场景下的的zip包是可控的，如通过文件上传等功能

```
package zip;

import org.springframework.core.io.DefaultResourceLoader;
import org.springframework.core.io.Resource;
import org.springframework.core.io.ResourceLoader;
import org.springframework.integration.support.MessageBuilder;
import org.springframework.integration.zip.transformer.UnZipTransformer;
import org.springframework.messaging.Message;

import java.io.File;
import java.io.InputStream;

public class Zip3 {
    private static ResourceLoader resourceLoader = new DefaultResourceLoader();

    public static void main(String[] args) {
        final Resource evilResource = resourceLoader.getResource("classpath:poc.zip");
        try{
            InputStream evilIS = evilResource.getInputStream();
            Message<InputStream> evilMessage = MessageBuilder.withPayload(evilIS).build();
            UnZipTransformer unZipTransformer = new UnZipTransformer();
            unZipTransformer.transform(evilMessage);
        }catch (Exception e){
            System.out.println(e);
        }
    }
}
```

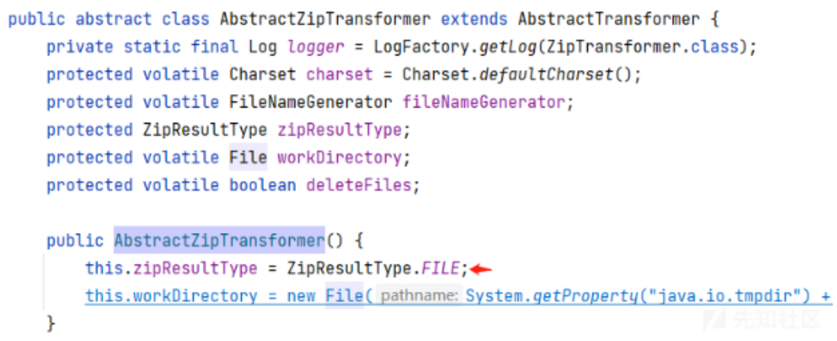
跟进 `org.springframework.integration.zip.transformer.UnZipTransformer#UnZipTransformer` 构造方法



```
public class UnZipTransformer extends AbstractZipTransformer {
    private static final Log logger = LogFactory.getLog(UnZipTransformer.class);
    private volatile boolean expectSingleResult = false;

    public UnZipTransformer() {
    }
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129112113-fc1205de-9f83-1.png)

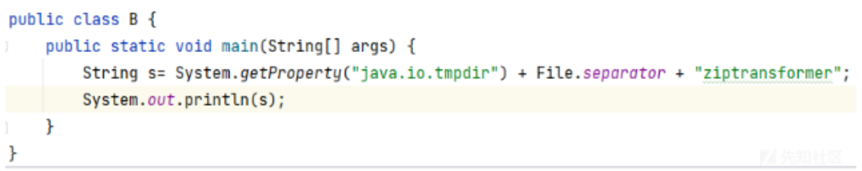
跟进 `org.springframework.integration.zip.transformer.AbstractZipTransformer#AbstractZipTransformer` 构造方法



```
public abstract class AbstractZipTransformer extends AbstractTransformer {
    private static final Log logger = LogFactory.getLog(ZipTransformer.class);
    protected volatile Charset charset = Charset.defaultCharset();
    protected volatile FileNameGenerator fileNameGenerator;
    protected ZipResultType zipResultType;
    protected volatile File workDirectory;
    protected volatile boolean deleteFiles;

    public AbstractZipTransformer() {
        this.zipResultType = ZipResultType.FILE;
        this.workDirectory = new File( pathname: System.getProperty("java.io.tmpdir") +
    }
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129112228-2890d31a-9f84-1.png)

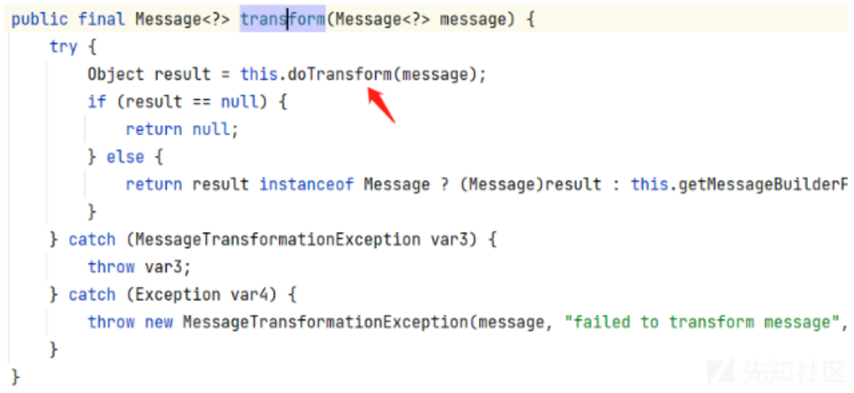
初始化了 `zipResultType` 和 `workDirectory` 属性，前者为 `ZipResultType.FILE` ，后续会用到此值，而 `workDirectory` 默认值为 `new File(System.getProperty("java.io.tmpdir") + File.separator + "ziptransformer")` 后续也会使用到该值。在我的测试环境下， `System.getProperty("java.io.tmpdir") + File.separator + "ziptransformer"` 如下：



```
public class B {
    public static void main(String[] args) {
        String s= System.getProperty("java.io.tmpdir") + File.separator + "ziptransformer";
        System.out.println(s);
    }
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129112332-4f18146c-9f84-1.png)

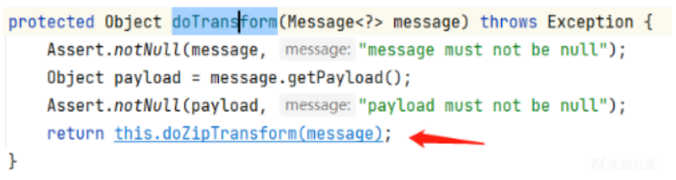
创建完 `UnZipTransformer` 后，执行 `org.springframework.integration.transformer.AbstractTransformer#transform` 方法



```java
public final Message<?> transform(Message<?> message) {
    try {
        Object result = this.doTransform(message);
        if (result == null) {
            return null;
        } else {
            return result instanceof Message ? (Message)result : this.getMessageBuilderF
        }
    } catch (MessageTransformationException var3) {
        throw var3;
    } catch (Exception var4) {
        throw new MessageTransformationException(message, "failed to transform message",
    }
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129112630-b9261732-9f84-1.png)

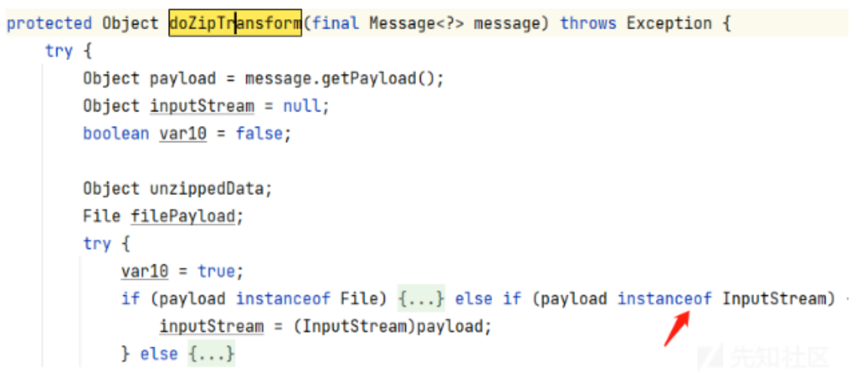
其中 `message` 参数值是 `zip` 文件读取流，继续跟进 `org.springframework.integration.zip.transformer.AbstractZipTransformer#doTransform`



```java
protected Object doTransform(Message<?> message) throws Exception {
    Assert.notNull(message,  message: "message must not be null");
    Object payload = message.getPayload();
    Assert.notNull(payload,  message: "payload must not be null");
    return this.doZipTransform(message);
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129112709-d03c0cba-9f84-1.png)

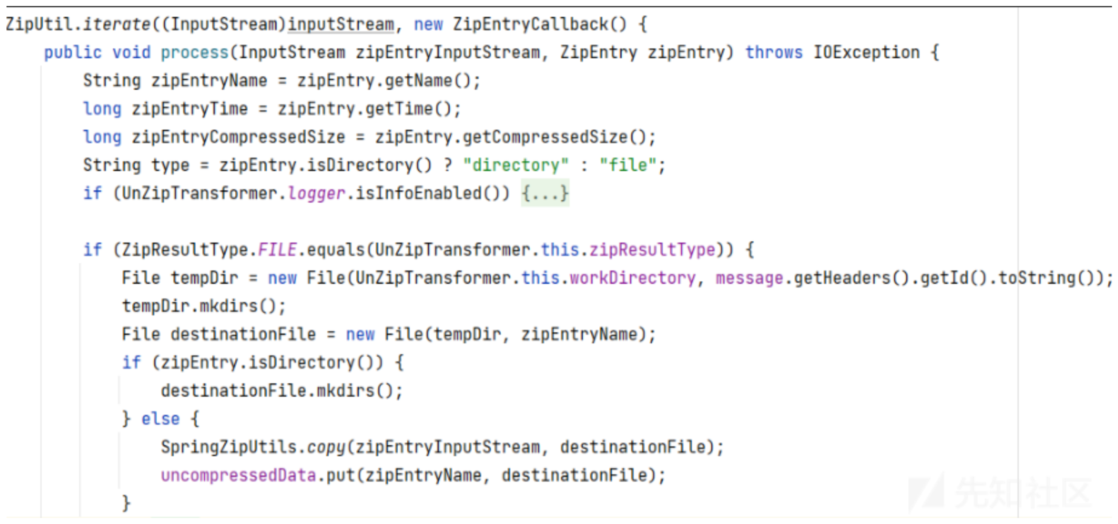
继续跟进 `org.springframework.integration.zip.transformer.UnZipTransformer#doZipTransform`



```java
protected Object doZipTransform(final Message<?> message) throws Exception {
    try {
        Object payload = message.getPayload();
        Object inputStream = null;
        boolean var10 = false;

        Object unzippedData;
        File filePayload;
        try {
            var10 = true;
            if (payload instanceof File) {...} else if (payload instanceof InputStream)
                inputStream = (InputStream)payload;
            } else {...}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129112740-e293ed74-9f84-1.png)

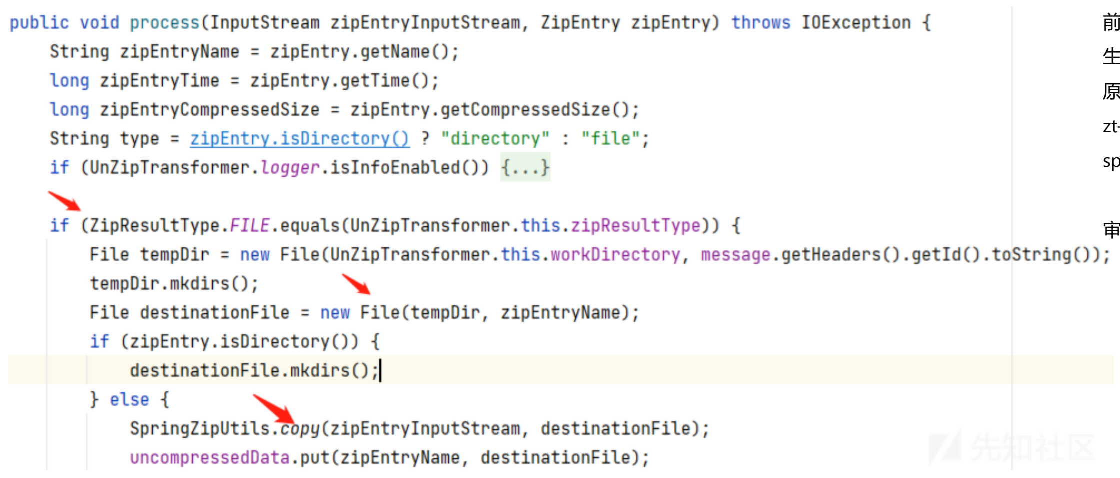
继续跟进



```java
ZipUtil.iterate((InputStream)inputStream, new ZipEntryCallback() {
    public void process(InputStream zipEntryInputStream, ZipEntry zipEntry) throws IOException {
        String zipEntryName = zipEntry.getName();
        long zipEntryTime = zipEntry.getTime();
        long zipEntryCompressedSize = zipEntry.getCompressedSize();
        String type = zipEntry.isDirectory() ? "directory" : "file";
        if (UnZipTransformer.logger.isInfoEnabled()) {...}

        if (ZipResultType.FILE.equals(UnZipTransformer.this.zipResultType)) {
            File tempDir = new File(UnZipTransformer.this.workDirectory, message.getHeaders().getId().toString());
            tempDir.mkdirs();
            File destinationFile = new File(tempDir, zipEntryName);
            if (zipEntry.isDirectory()) {
                destinationFile.mkdirs();
            } else {
                SpringZipUtils.copy(zipEntryInputStream, destinationFile);
                uncompressedData.put(zipEntryName, destinationFile);
            }
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129112803-f092e7a4-9f84-1.png)

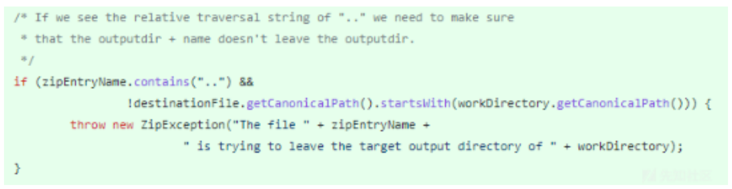
调用了 `zt-zip` 的 `api` ，只不过 `spring integration zip` 在这里自己创建了一个 `ZipEntryCallback` 匿名对象，最后会调用此匿名对象的 `process` 方法



```java
public void process(InputStream zipEntryInputStream, ZipEntry zipEntry) throws IOException {
    String zipEntryName = zipEntry.getName();
    long zipEntryTime = zipEntry.getTime();
    long zipEntryCompressedSize = zipEntry.getCompressedSize();
    String type = zipEntry.isDirectory() ? "directory" : "file";
    if (UnZipTransformer.logger.isInfoEnabled()) {...}

    if (ZipResultType.FILE.equals(UnZipTransformer.this.zipResultType)) {
        File tempDir = new File(UnZipTransformer.this.workDirectory, message.getHeaders().getId().toString());
        tempDir.mkdirs();
        File destinationFile = new File(tempDir, zipEntryName);
        if (zipEntry.isDirectory()) {
            destinationFile.mkdirs();
        } else {
            SpringZipUtils.copy(zipEntryInputStream, destinationFile);
            uncompressedData.put(zipEntryName, destinationFile);
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129112902-1369cf7c-9f85-1.png)

没有任何过滤，导致zip slip发生。
修复方案如下：https://github.com/spring-projects/spring-integration-extensions/commit/a5573eb232ff85199ff9bb28993df715d9a19a25 (https://github.com/spring-projects/spring-integration-extensions/commit/a5573eb232ff85199ff9bb28993df715d9a19a25)

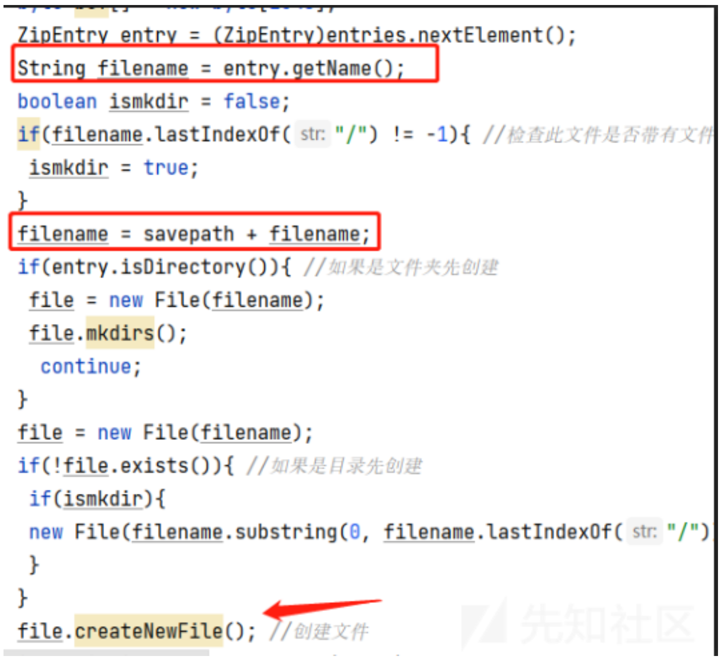


```java
/* If we see the relative traversal string of ".." we need to make sure
 * that the outputdir + name doesn't leave the outputdir.
 */
if (zipEntryName.contains("..") &&
                !destinationFile.getCanonicalPath().startsWith(workDirectory.getCanonicalPath())) {
        throw new ZipException("The file " + zipEntryName +
                        " is trying to leave the target output directory of " + workDirectory);
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129113328-b1e6b62e-9f85-1.png)

审计实战

项目地址：https://gitee.com/RainyGao/DocSys (https://gitee.com/RainyGao/DocSys)
在 `com.DocSystem.controller.BaseController#unZip` 方法中存在如下代码片段

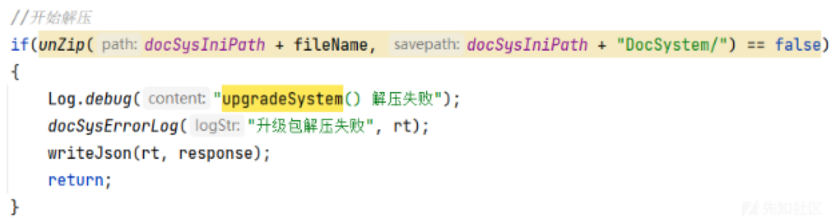


```java
ZipEntry entry = (ZipEntry)entries.nextElement();
String filename = entry.getName();
boolean ismkdir = false;
if(filename.lastIndexOf( str: "/") != -1){ //检查此文件是否带有文件
 ismkdir = true;
}
filename = savepath + filename;
if(entry.isDirectory()){ //如果是文件夹先创建
 file = new File(filename);
 file.mkdirs();
  continue;
}
file = new File(filename);
if(!file.exists()){ //如果是目录先创建
 if(ismkdir){
 new File(filename.substring(0, filename.lastIndexOf( str: "/")
 }
}
file.createNewFile(); //创建文件
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129202450-ed7470d8-9fcf-1.png)

其中 `entry.getName()` 的值是可控的，通过 `../` 可以将恶意jsp文件写到web根目录。
寻找触发点，发现在 `com.DocSystem.controller.ManageController#upgradeSystem` 方法中触发了 `com.DocSystem.controller.BaseController#unZip` 方法

关键代码如下：


```
//开始解压
if(unZip( path: docSysIniPath + fileName,  savepath: docSysIniPath + "DocSystem/") == false)
{
    Log.debug( content: "upgradeSystem() 解压失败");
    docSysErrorLog( logStr: "升级包解压失败", rt);
    writeJson(rt, response);
    return;
}
```

(https://xzfile.aliyuncs.com/media/upload/picture/20230129203146-e4ec79fa-9fd0-1.png)
具体漏洞复现可参考：https://gitee.com/RainyGao/DocSys/issues/I65IYU (https://gitee.com/RainyGao/DocSys/issues/I65IYU)

打赏　关注 | 4　点击收藏 | 6

上一篇： 6.bWAPP XSS (/t/12080)　下一篇： 记一次金融站点的验签破解实战 (/t/12082)

1 条回复

t*枫 (/u/73388) **2023-09-14 11:13:51 来自广东**

不好意思打扰博主了，我发现您写的文章都很不错，可以转载您主页里的文章到opensnn吗？我会在转载的文章下标记出处和作者。请博主放心，在还没有博主您的同意前我不会进行转载。

0　回复Ta

登录 (https://account.aliyun.com/login/login.htm?oauth_callback=https%3A%2F%2Fxz.aliyun.com%2Ft%2F12081&from_type=xianzhi) 后跟帖

先知社区

现在登录 (https://account.aliyun.com/l

社区小黑板 (/notice)

年度贡献榜　月度贡献榜

OpenSCA社区 (/u/645...　2