CVE-2020-10189/Zoho ManageEngine Desktop Central 10 反序列化远程代码执行 - 先知社区

> 先知社区，先知安全技术社区

漏洞描述

2020 年 3 月 6 日, 有安全研究人员 `steventseeley` 在 Twitter 上发布了 Zoho 企业产品 `Zoho ManageEngine Desktop Central` 中的反序列化远程代码执行漏洞. 利用该漏洞可直接控制安装该产品的服务器, 该产品为 Windows 系统上运行的一种端点管理解决方案, 客户可用它管理网络中的设备, 例如：Android 手机, Unix 服务器以及 Windows 工作站等. 它往往充当公司内部的中央服务器, 帮助系统管理员推送更新，远程控制系统，锁定设备，控制访问权限等。

影响版本

Zoho ManageEngine Desktop Central < 10.0.479

漏洞编号

CVE-2020-10189

威胁等级

高危

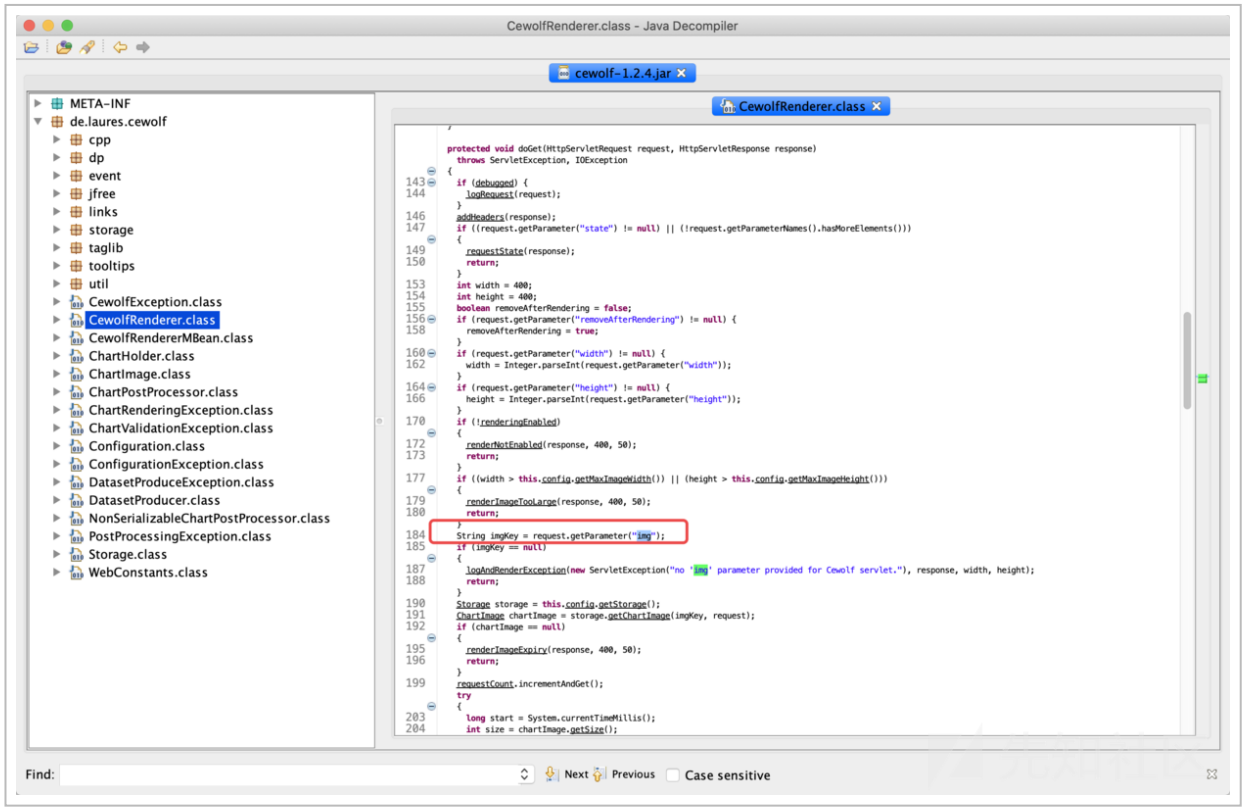
漏洞分析

1. 寻找反序列化点

我们进入 `DesktopCentral_Server/webapps/DesktopCentral/WEB-INF/` 目录提取 `web.xml`，可以看到一个名为 `CewolfServlet` 的 `servlet`.

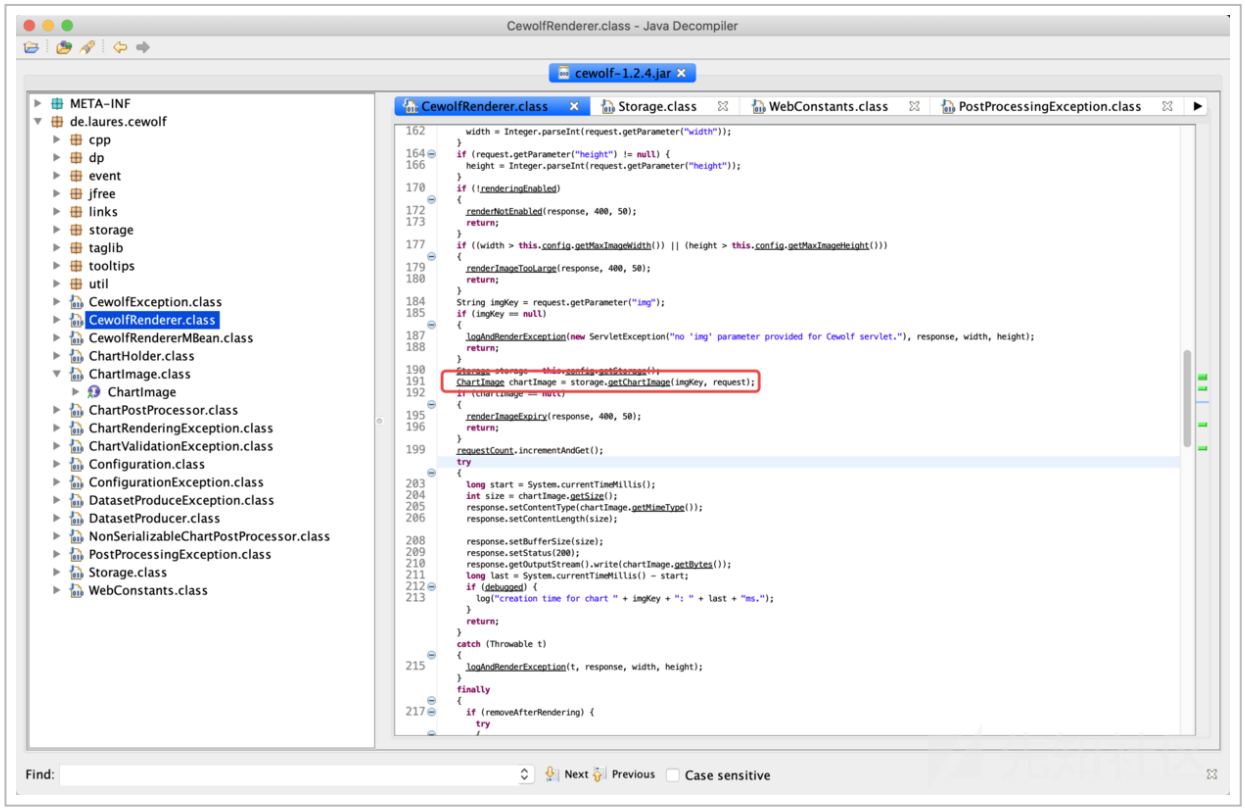
(https://xzfile.aliyuncs.com/media/upload/picture/20200315111817-9d0a9050-666b-1.png)

该 `servlet` 对应的 `class` 为 `de.laures.cewolf.CewolfRenderer`，该类存在于 `DesktopCentral_Server/lib/cewolf-1.2.4.jar`
该类中的 get 方法使用 `img` 参数接受 `imgKey` 的值


CewolfRenderer.class - Java Decompiler
cewolf-1.2.4.jar
META-INF
de.laures.cewolf
cpp
dp
event
jfree
links
storage
taglib
tooltips
util
CewolfException.class
CewolfRenderer.class
CewolfRendererMBean.class
ChartHolder.class
ChartImage.class
ChartPostProcessor.class
ChartRenderingException.class
ChartValidationException.class
Configuration.class
ConfigurationException.class
DatasetProduceException.class
DatasetProducer.class
NonSerializableChartPostProcessor.class
PostProcessingException.class
Storage.class
WebConstants.class
CewolfRenderer.class
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
if (debugged) {
logRequest(request);
}
addHeaders(response);
if ((request.getParameter("state") != null) || (!request.getParameterNames().hasMoreElements()))
{
requestState(response);
return;
}
int width = 400;
int height = 400;
boolean removeAfterRendering = false;
if (request.getParameter("removeAfterRendering") != null) {
removeAfterRendering = true;
}
if (request.getParameter("width") != null) {
width = Integer.parseInt(request.getParameter("width"));
}
if (request.getParameter("height") != null) {
height = Integer.parseInt(request.getParameter("height"));
}
if (!renderingEnabled)
{
renderNotEnabled(response, 400, 50);
return;
}
if ((width > this.config.getMaxImageWidth()) || (height > this.config.getMaxImageHeight()))
{
renderImageTooLarge(response, 400, 50);
return;
}
String imgKey = request.getParameter("img");
if (imgKey == null)
{
logAndRenderException(new ServletException("no 'img' parameter provided for Cewolf servlet."), response, width, height);
return;
}
Storage storage = this.config.getStorage();
ChartImage chartImage = storage.getChartImage(imgKey, request);
if (chartImage == null)
{
renderImageExpiry(response, 400, 50);
return;
}
requestCount.incrementAndGet();
try
{
long start = System.currentTimeMillis();
int size = chartImage.getSize();
Find:
Next
Previous
Case sensitive

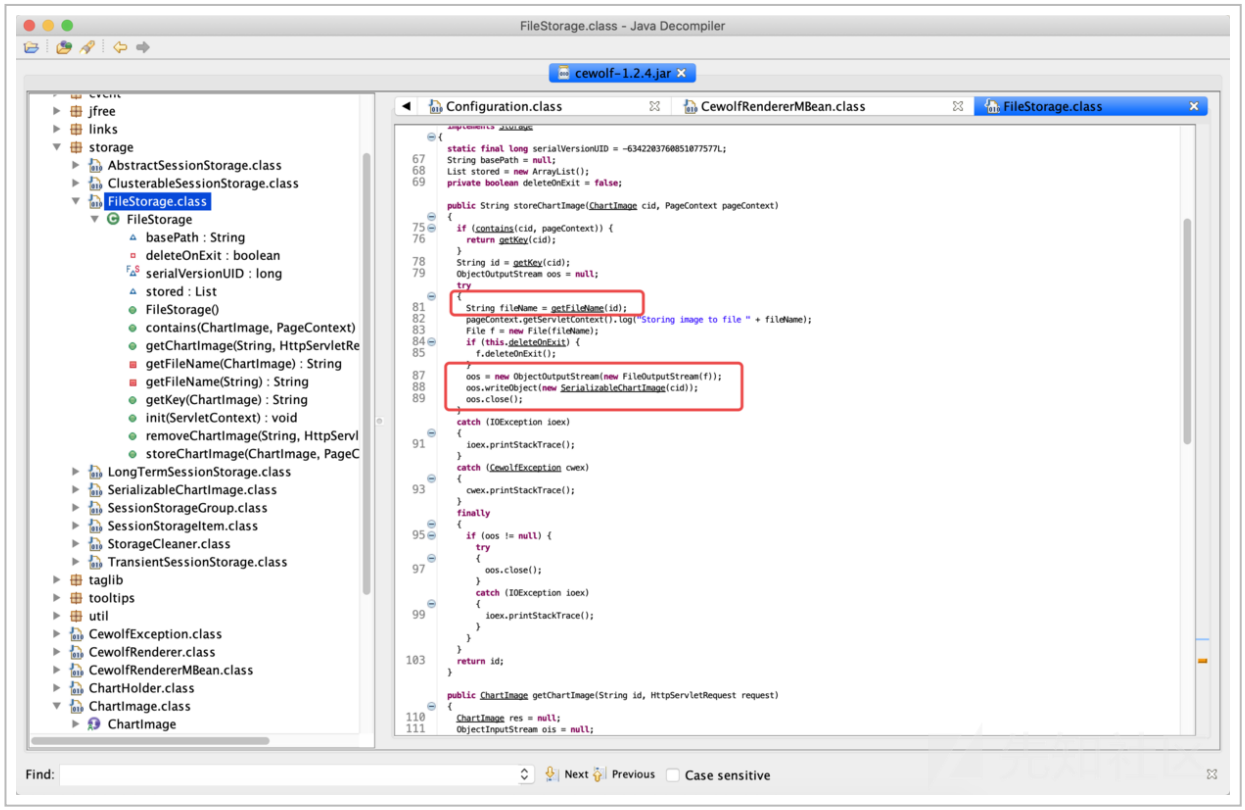
(https://xzfile.aliyuncs.com/media/upload/picture/20200315111829-a474eac0-666b-1.png)

imgKey 将会被 `Storage` 类中的 `getChartImage` 方法调用

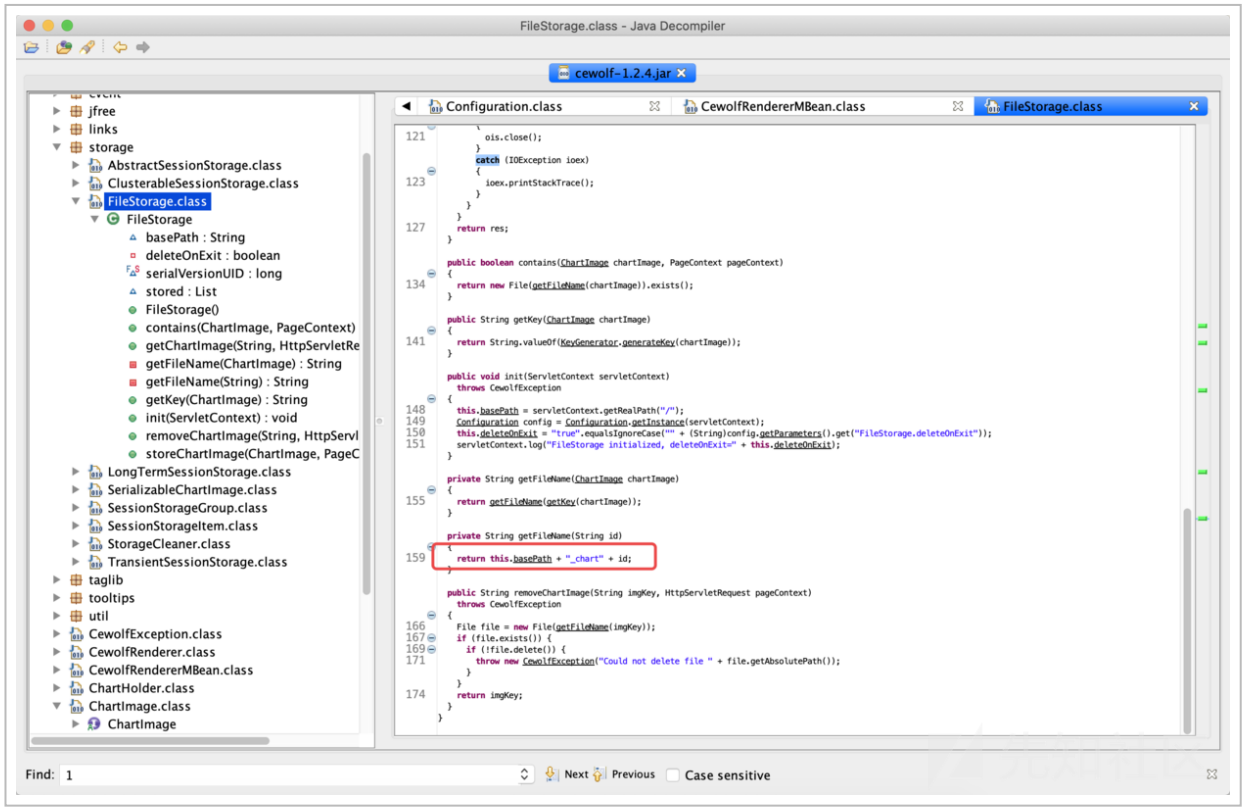


(https://xzfile.aliyuncs.com/media/upload/picture/20200315111840-aaee194e-666b-1.png)

在该 jar 包中存在一个 `FileStorage` 类基于 `Storage` 类实现接口的类. 在其内部存在一个 `storeChartImage` 方法, 该方法直接将接收到的文件流存储在本地. 该方法内部调用一个 `getFileName` 类用于拼接 base 路径.



(https://xzfile.aliyuncs.com/media/upload/picture/20200315111851-b1bb9b20-666b-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200315111900-b732777c-666b-1.png)

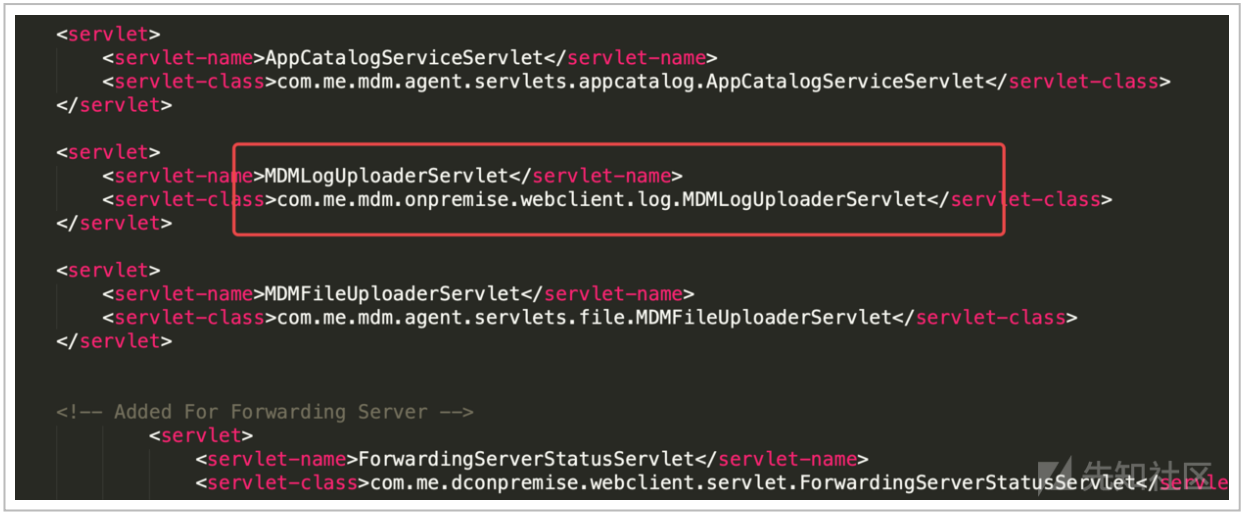
在查看 `storeChartImage` 方法的同时, 我注意到还有一个获取文件的方法 `getChartImage` .

(https://upload-images.jianshu.io/upload_images/5350990-1d357b29f9262c20.png?imageMogr2/auto-orient/strip%7CimageView2/2/w/1240)

清晰可见的是它执行了一个非常危险的操作, 直接让 `ObjectInputStream` 执行了 `readObject` 方法. 可见这就是漏洞的触发点. 那么问题来了, 有了反序列化的点, 序列化文件从哪里上传?

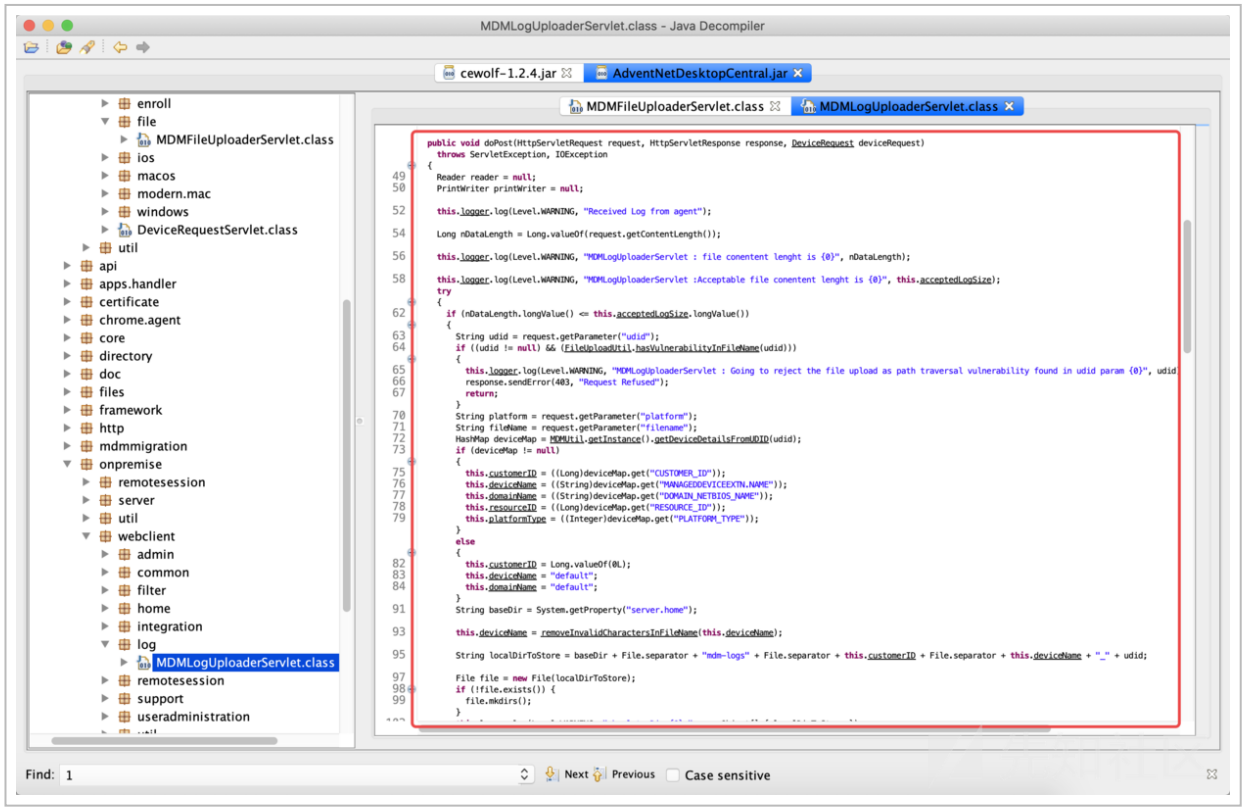
2. 寻找文件上传点

于是我再次检索 `web.xml` , 发现了一个具有上传功能的 `servlet` .



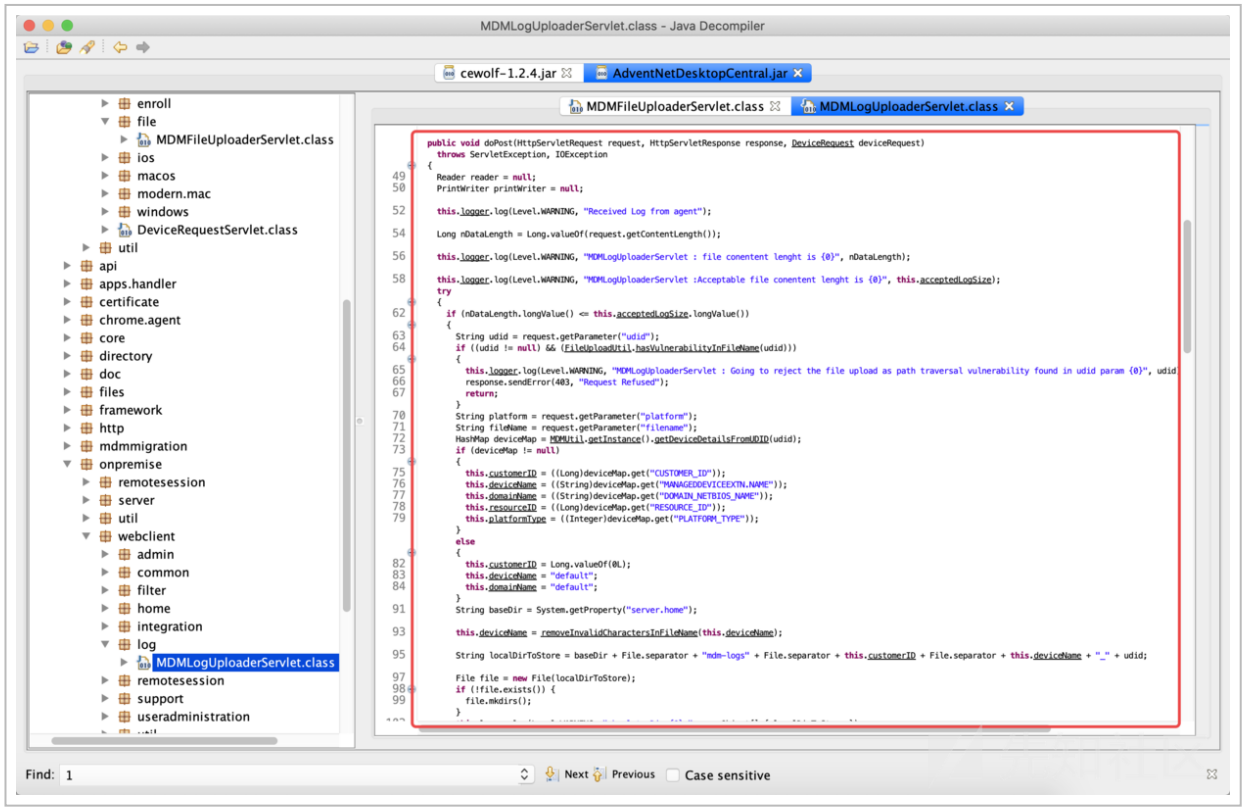
(https://xzfile.aliyuncs.com/media/upload/picture/20200315112057-fcbbfec6-666b-1.png)

该 `servlet` 对应 `DesktopCentral_Server/lib/AdventNetDesktopCentral.jar` 中的 `com.me.mdm.agent.servlets.file.MDMFileUploaderServlet` .

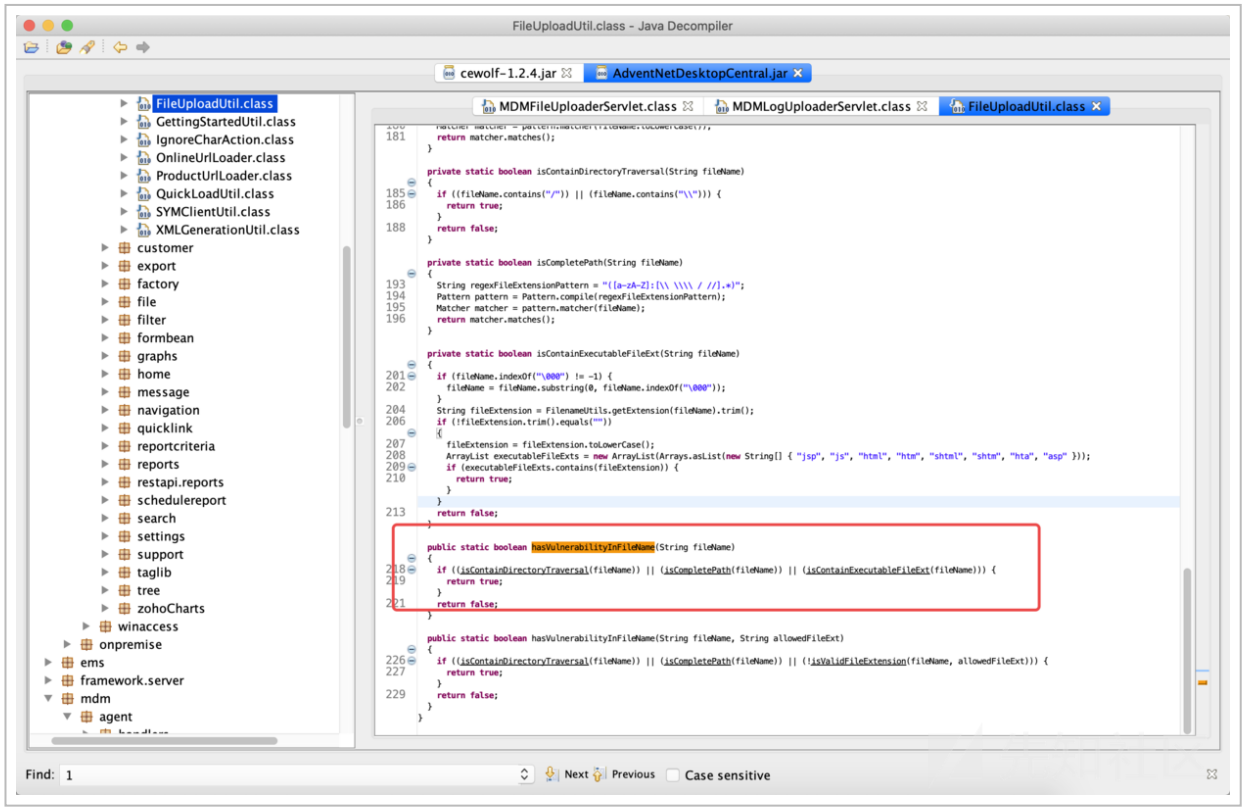


(https://xzfile.aliyuncs.com/media/upload/picture/20200315112136-13db1f24-666c-1.png)

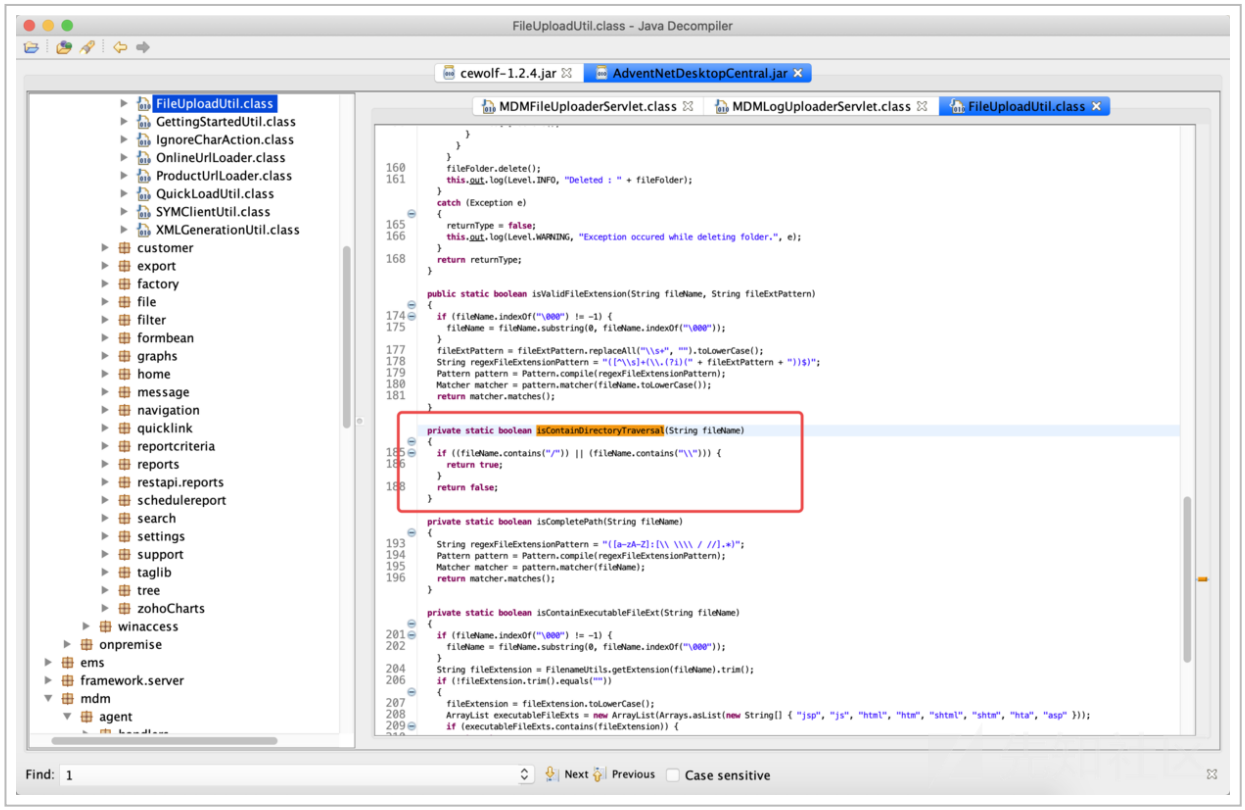
在该类中存在 `Post` 方法, 可见 `udid` 和 `filename` 为可控点, 我们完全可以自己传值控制. 当然 `udid` 其实也做了三种安全检查, 但并不影响目录穿越的产生.



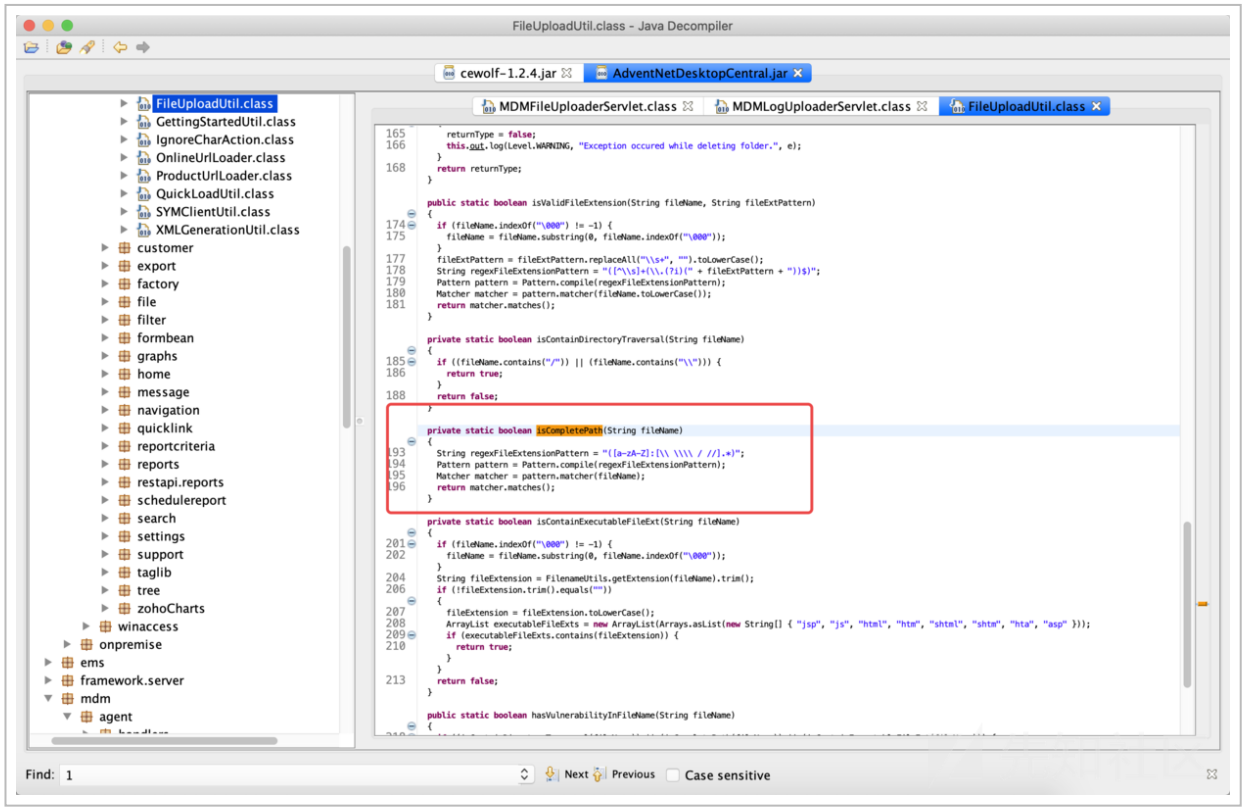
(https://xzfile.aliyuncs.com/media/upload/picture/20200315113513-fad2ab80-666d-1.png)



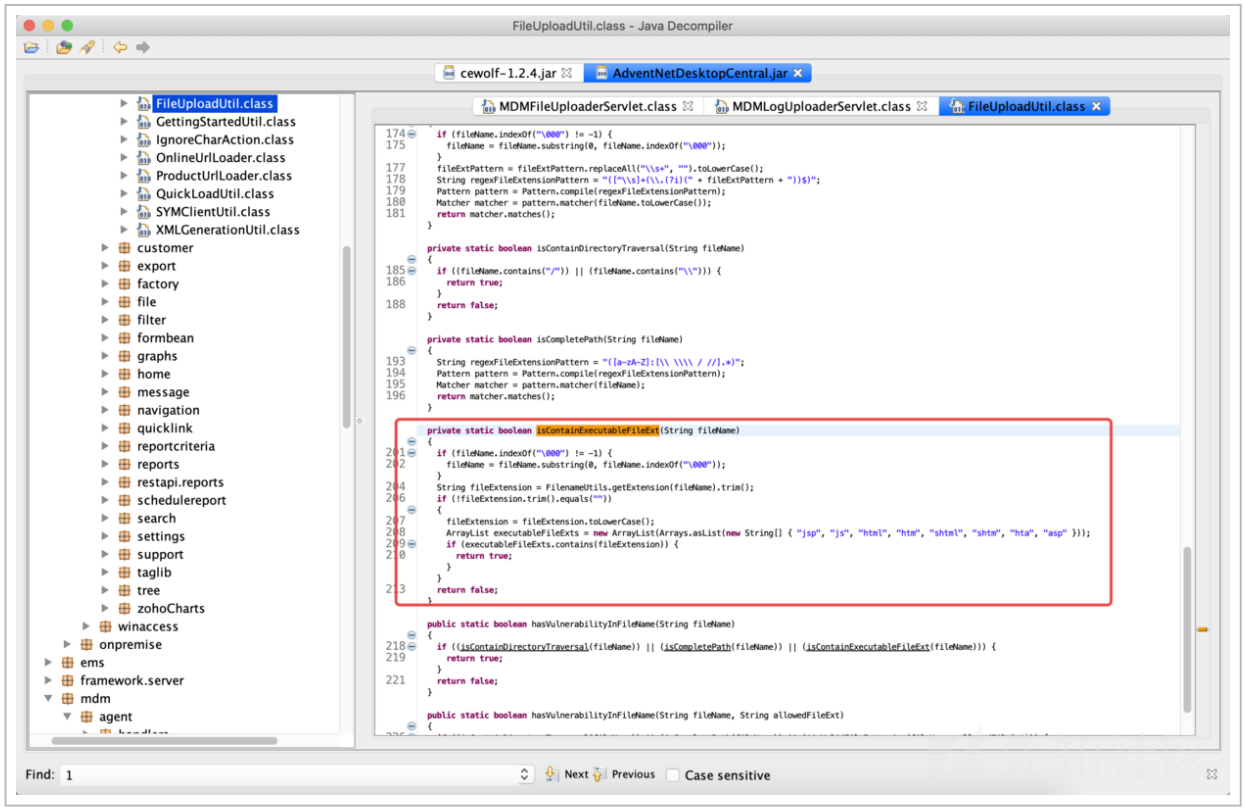
(https://xzfile.aliyuncs.com/media/upload/picture/20200315112303-47dafbe6-666c-1.png)



(https://xzfile.aliyuncs.com/media/upload/picture/20200315112316-4fa174a4-666c-1.png)

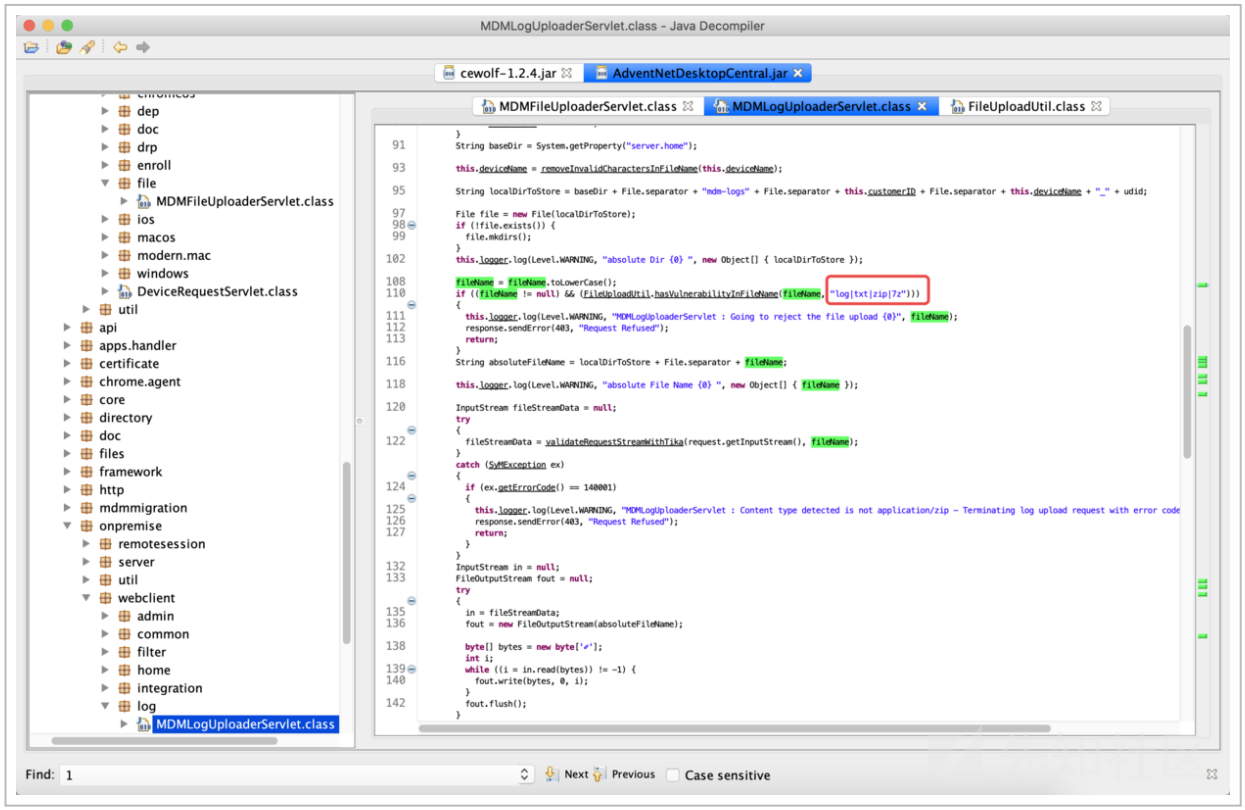


(https://xzfile.aliyuncs.com/media/upload/picture/20200315112326-55a5d340-666c-1.png)



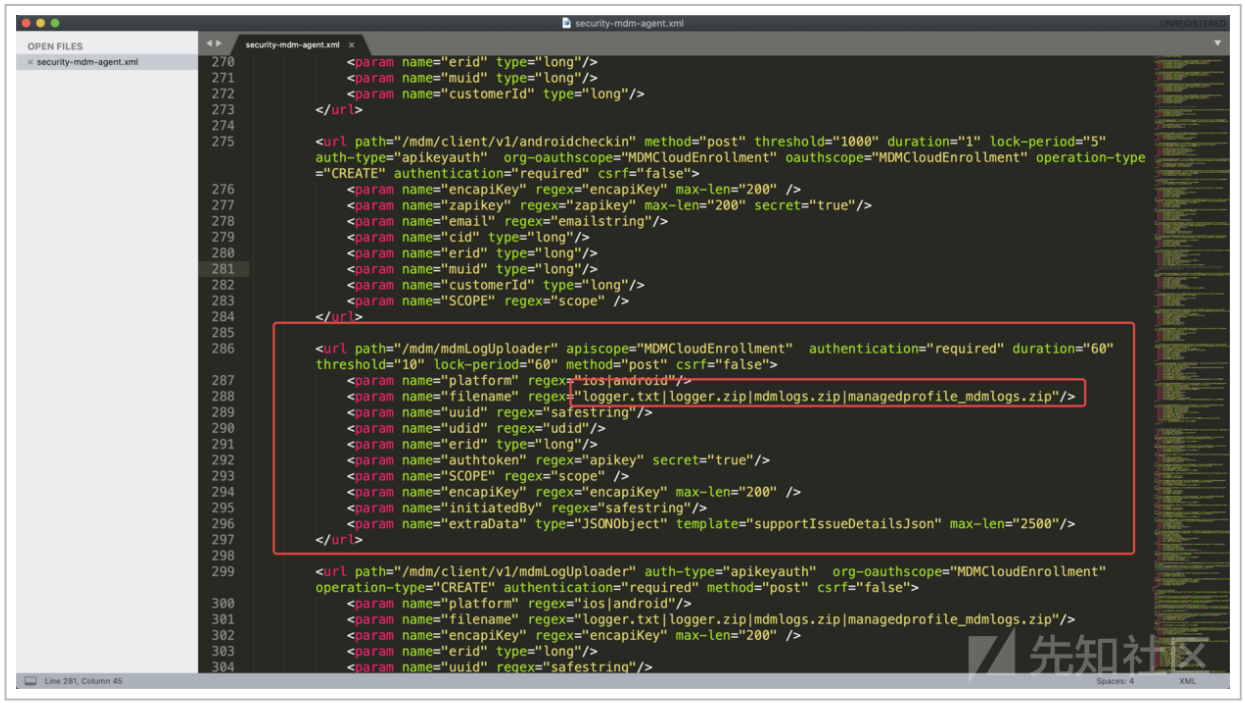
(https://xzfile.aliyuncs.com/media/upload/picture/20200315112334-5a8696ba-666c-1.png)

而 `filename` 被做了后缀名限制, 只允许为 `log|txt|zip|7z` 其中一种



(https://xzfile.aliyuncs.com/media/upload/picture/20200315112512-9470b086-666c-1.png)

其实在 `security-mdm-agent.xml` 做了进一步限制



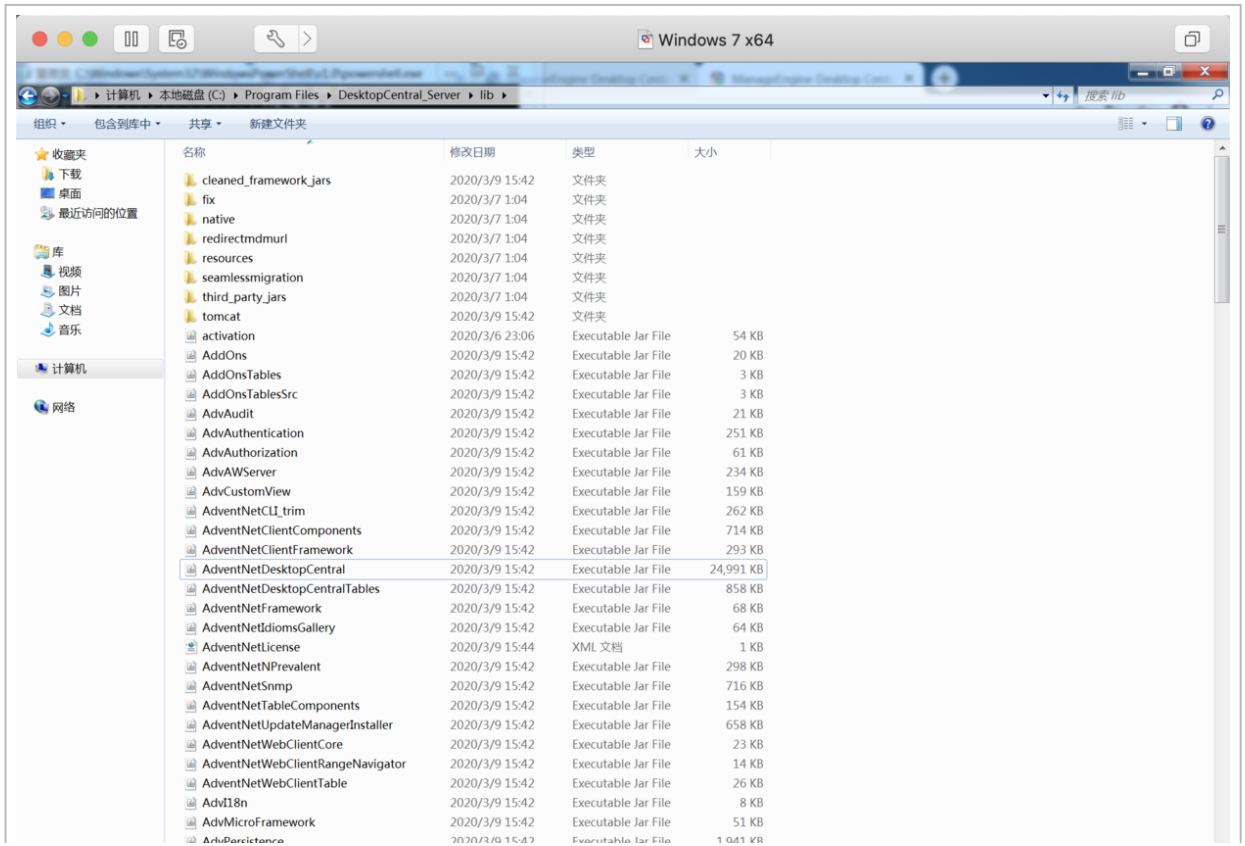
(https://xzfile.aliyuncs.com/media/upload/picture/20200315112529-9f0c40dc-666c-1.png)

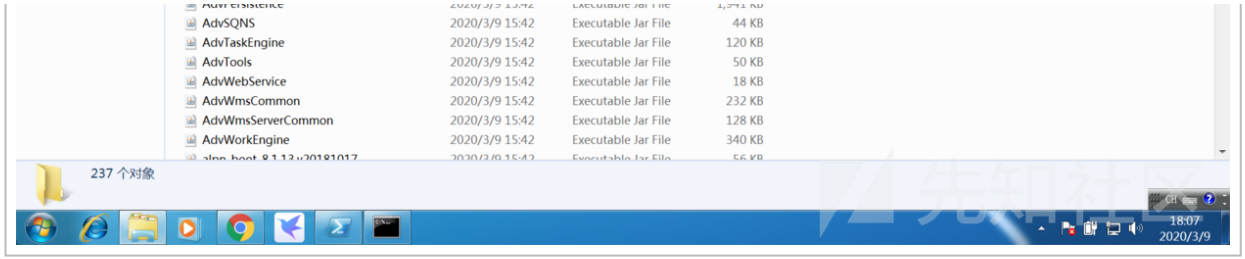
1.png)

只允许完整文件名称为 `logger.txt|logger.zip|mdmlogs.zip|managedprofile_mdmlogs.zip` . 不过这丝毫不影响我们上传恶意的序列化文件.

3. 寻找序列化可用的 gadgets

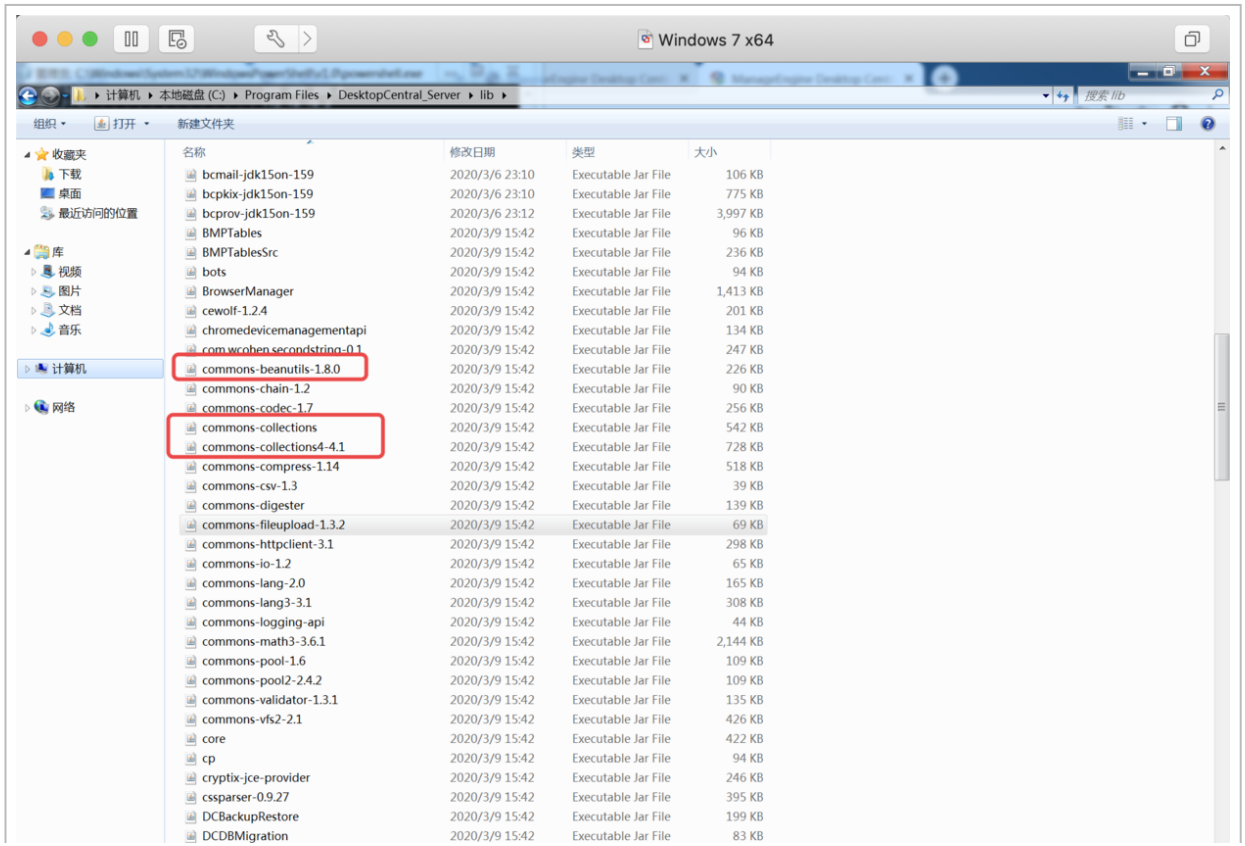
我们进入 `DesktopCentral_Server\lib` 目录, 寻找有漏洞的 jar.

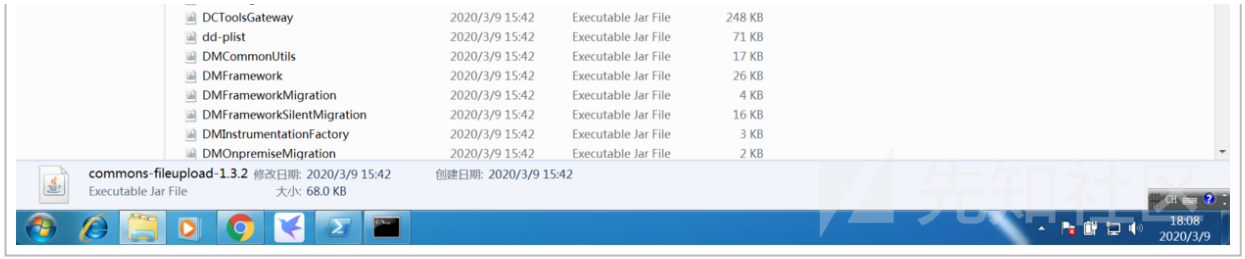




(https://xzfile.aliyuncs.com/media/upload/picture/20200315112637-c79fc924-666c-1.png)

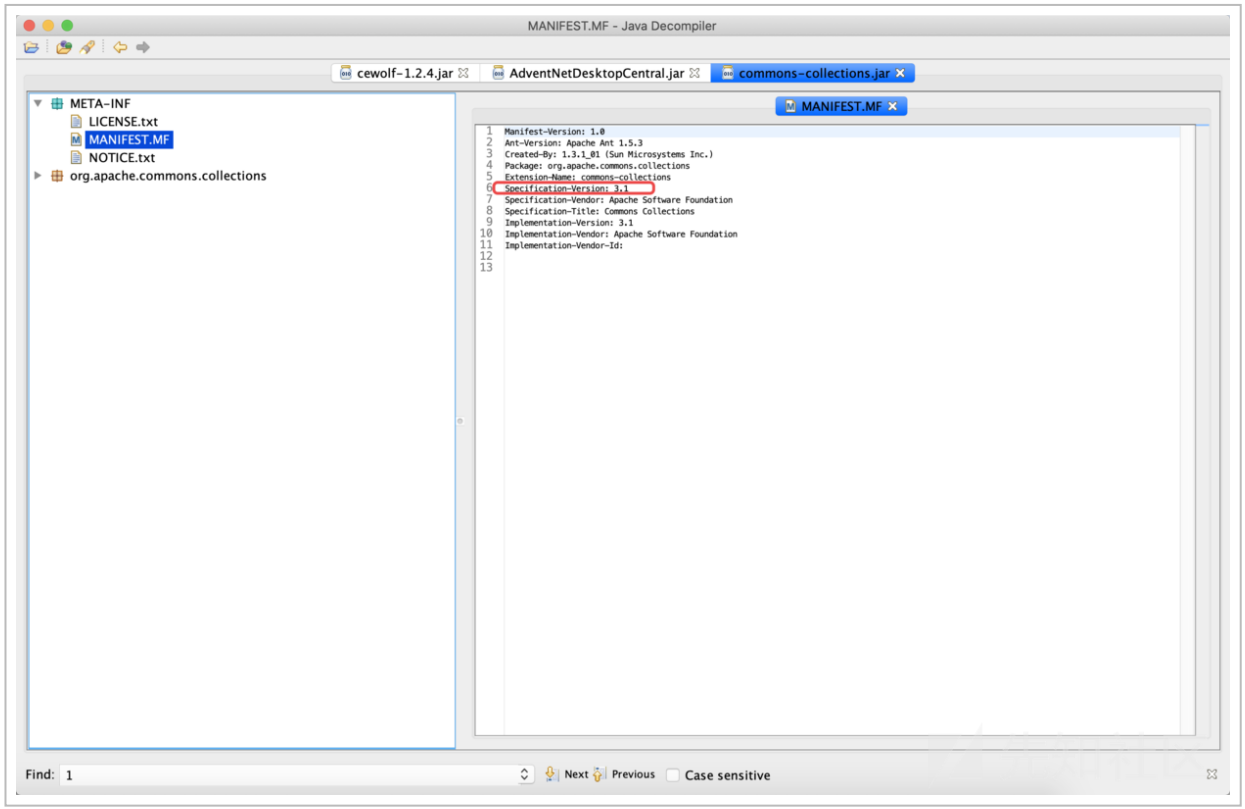
可见我们发现了 `commons-collections.jar` 、 `commons-beanutils-1.8.0.jar`





(https://xzfile.aliyuncs.com/media/upload/picture/20200315112701-d59fd7d0-666c-1.png)

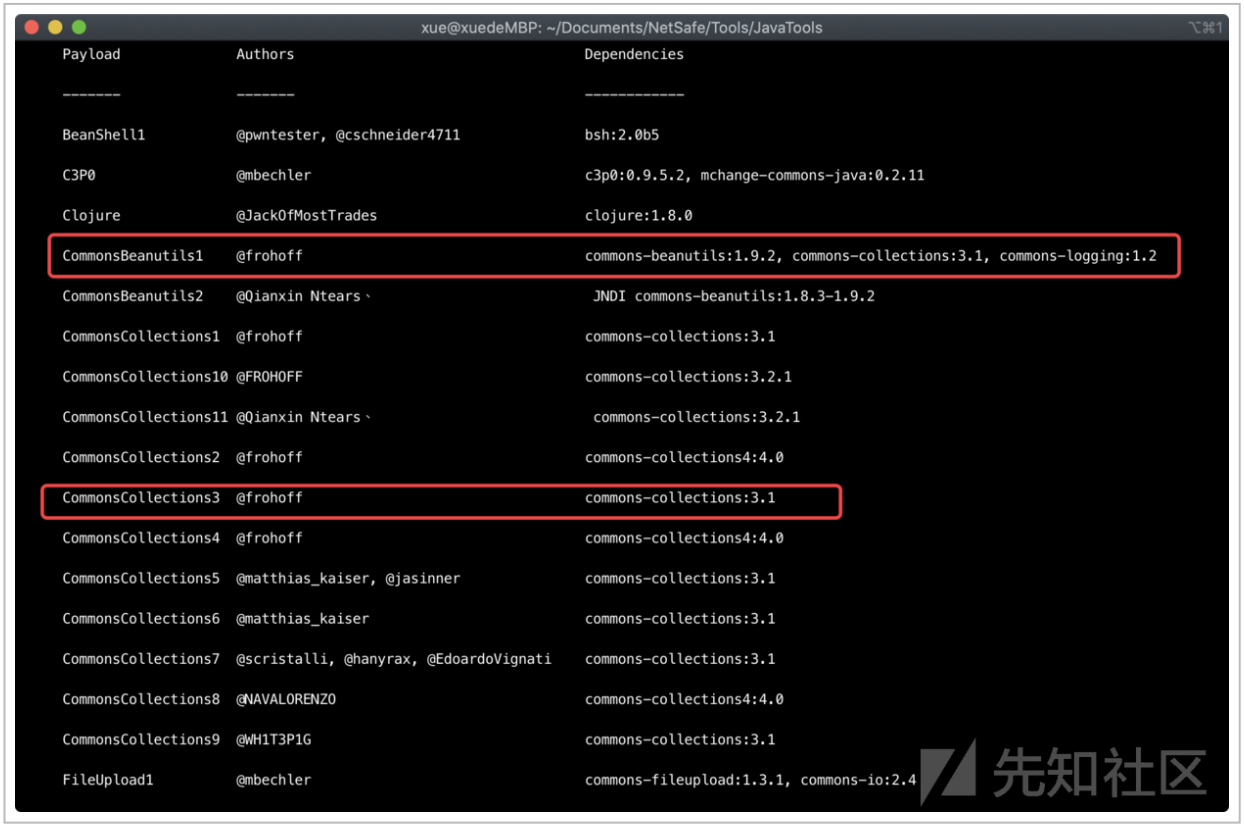
`commons-collections.jar` 不确定具体版本, 我们查看一下版本



(https://xzfile.aliyuncs.com/media/upload/picture/20200315112715-de17be14-666c-1.png)

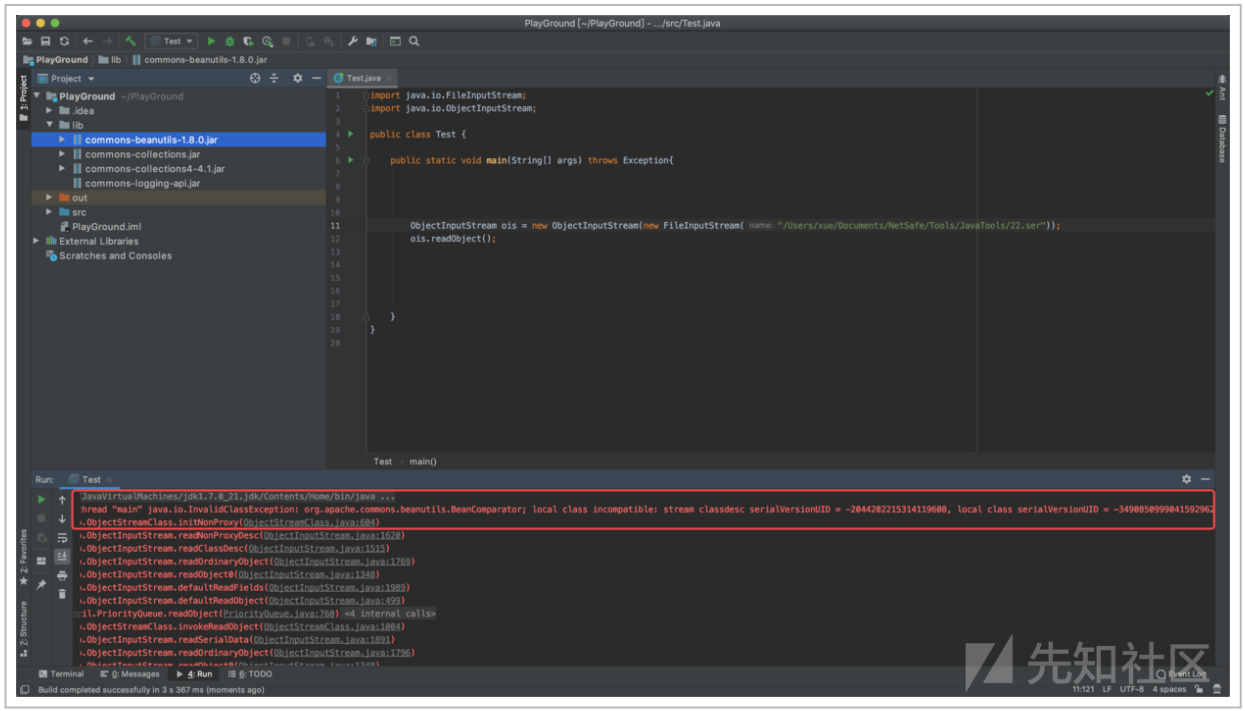
太完美了它是 `commons-collections:3.1` . 可见我们可以利用 `CommonsBeanutils1` 和 `CommonsCollections3` 两条 gagdets. 当然还有 jdk 的两条 gagdets, `JDK7U21` 和 `JRE8U20` .

我们请出反序列化文件构建神器—— `ysoserial` .



(https://xzfile.aliyuncs.com/media/upload/picture/20200315112751-f3a208d4-666c-1.png)

我们使用 `ysoserial` 中的两条 gadgets 构建序列化文件即可. 但是这里我们需要注意的是如果直接使用可能会反序列化失败,



(https://xzfile.aliyuncs.com/media/upload/picture/20200315112803-fa7a4806-666c-

1.png)

这是由于我们在打包 `ysoserial` 时默认的依赖版本是 `1.9.2`，
而 `Zoho ManageEngine Desktop Central` 自带的是 `commons-beanutils-1.8.0`，这将导致 `UID` 不同，从而造成反序列化失败. 我们只需要修改 `ysoserial` 的 `pom.xml` 中的 `commons-beanutils` 为 1.8 系列重新打包 `ysoserial` 即可

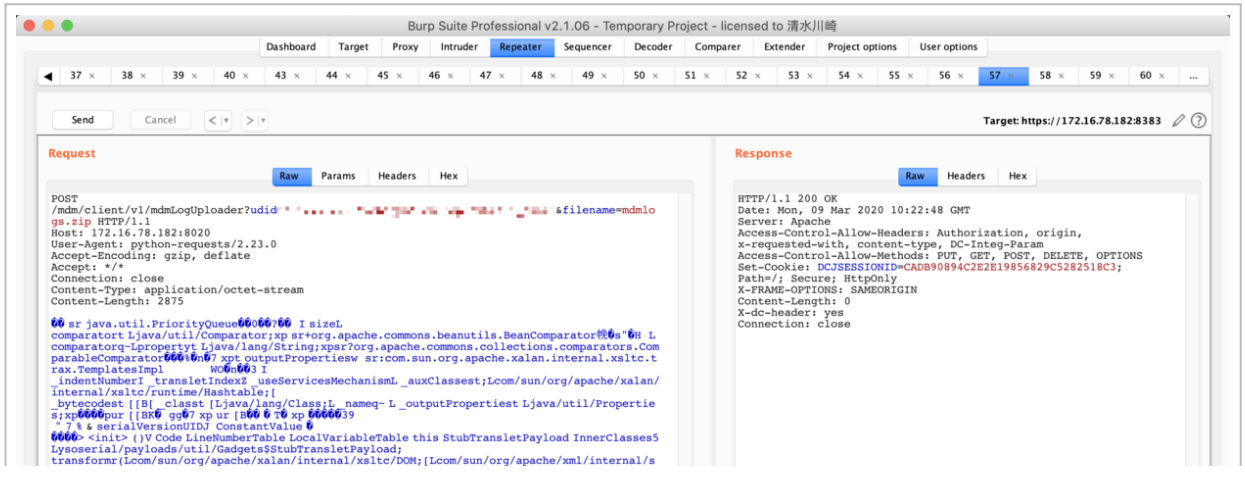
```
<dependency>
    <groupId>commons-collections</groupId>
    <artifactId>commons-collections</artifactId>
    <version>3.1</version>
</dependency>
<dependency>
    <groupId>org.beanshell</groupId>
    <artifactId>bsh</artifactId>
    <version>2.0b5</version>
</dependency>
<dependency>
    <groupId>commons-beanutils</groupId>
    <artifactId>commons-beanutils</artifactId>
    <version>1.8.3</version>
</dependency>
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-collections4</artifactId>
    <version>4.0</version>
```

```
<dependency>
    <groupId>org.codehaus.groovy</groupId>
    <artifactId>groovy</artifactId>
    <version>2.3.9</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>4.1.4.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-beans</artifactId>
```

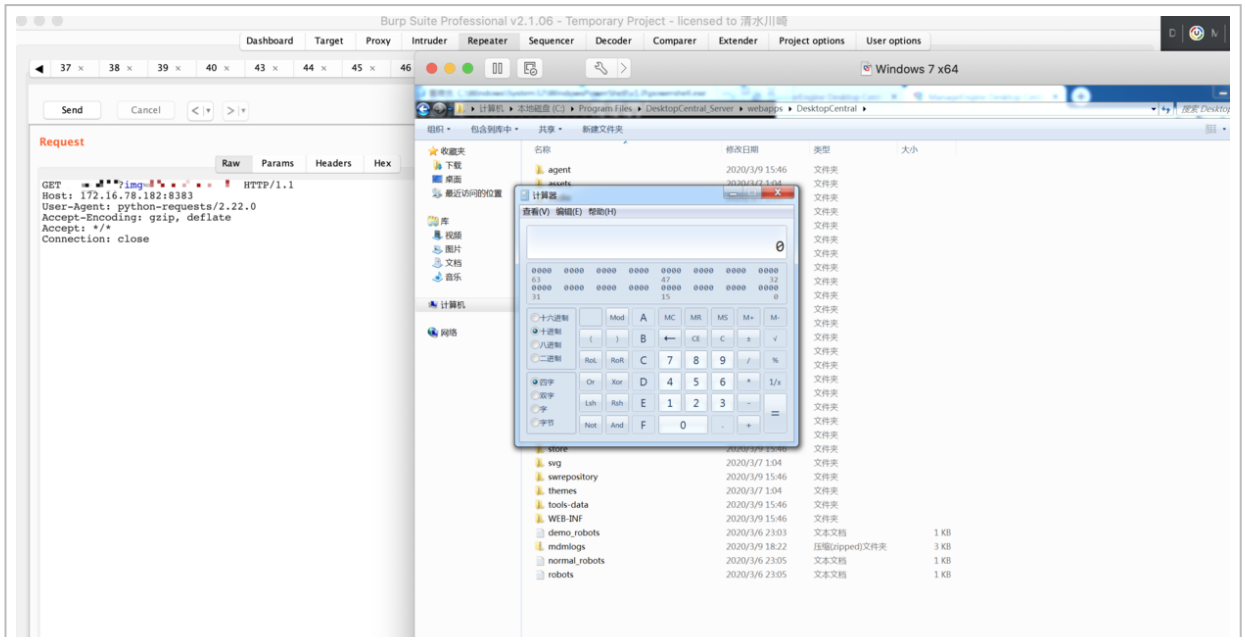
(https://xzfile.aliyuncs.com/media/upload/picture/20200315112839-10290534-666d-1.png)

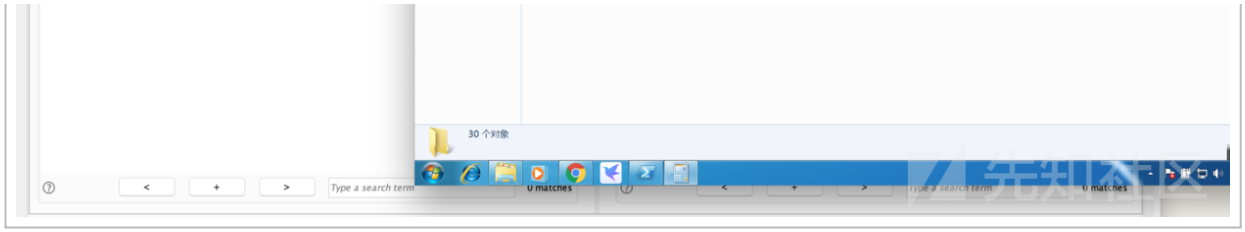
4. 效果演示

为了满足漏洞管理条例, 本文不直接放出 payload, 可自行参考文末的公开 payload



(https://xzfile.aliyuncs.com/media/upload/picture/20200315112857-1ae3f010-666d-1.png)





(https://xzfile.aliyuncs.com/media/upload/picture/20200315112907-20d914c8-666d-1.png)

Reference

https://nosec.org/home/detail/4211.html (https://nosec.org/home/detail/4211.html)
https://srcincite.io/pocs/src-2020-0011.py.txt (https://srcincite.io/pocs/src-2020-0011.py.txt)
https://nvd.nist.gov/vuln/detail/CVE-2020-10189 (https://nvd.nist.gov/vuln/detail/CVE-2020-10189)