

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` .

```
<servlet>
<servlet-name>CewolfServlet</servlet-name>
<servlet-class>de.laures.cewolf.CewolfRenderer</servlet-class>

<init-param>
  <param-name>debug</param-name>
  <param-value>>false</param-value>
</init-param>
<init-param>
  <param-name>overliburl</param-name>
  <param-value>/js/overlib.js</param-value>
</init-param>
<init-param>
  <param-name>storage</param-name>
  <param-value>de.laures.cewolf.storage.FileStorage</param-value>
</init-param>
```



(<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 x

CewolfRenderer.class x

- ▶ 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
 - ▶ DatasetProducerException.class
 - ▶ DatasetProducer.class
 - ▶ NonSerializableChartPostProcessor.class
 - ▶ PostProcessingException.class
 - ▶ Storage.class
 - ▶ WebConstants.class

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
{
    143 if (debugged) {
    144     logRequest(request);
    146 }
    147 addHeader(response);
    148 if ((request.getParameter("state") != null) || (request.getParameterNames().hasMoreElements()))
    149 {
    150     requestState(response);
    151     return;
    152 }
    153 int width = 400;
    154 int height = 400;
    155 boolean removeAfterRendering = false;
    156 if (request.getParameter("removeAfterRendering") != null) {
    157     removeAfterRendering = true;
    158 }
    159 if (request.getParameter("width") != null) {
    160     width = Integer.parseInt(request.getParameter("width"));
    161 }
    162 if (request.getParameter("height") != null) {
    163     height = Integer.parseInt(request.getParameter("height"));
    164 }
    165 if (!renderingEnabled)
    166 {
    167     renderNotEnabled(response, 400, 50);
    168     return;
    169 }
    170 if ((width > this.config.getMaxImageWidth()) || (height > this.config.getMaxImageHeight()))
    171 {
    172     renderImageTooLarge(response, 400, 50);
    173     return;
    174 }
    175 String imgKey = request.getParameter("img");
    176 if (imgKey == null)
    177 {
    178     logAndRenderException(new ServletException("no 'img' parameter provided for Cewolf servlet."), response, width, height);
    179     return;
    180 }
    181 Storage storage = this.config.getStorage();
    182 ChartImage chartImage = storage.getChartImage(imgKey, request);
    183 if (chartImage == null)
    184 {
    185     renderImageExpired(response, 400, 50);
    186     return;
    187 }
    188 requestCount.incrementAndGet();
    189 try
    190 {
    191     long start = System.currentTimeMillis();
    192     int size = chartImage.getSize();
    193 }
```

Find: Next Previous Case sensitive

33

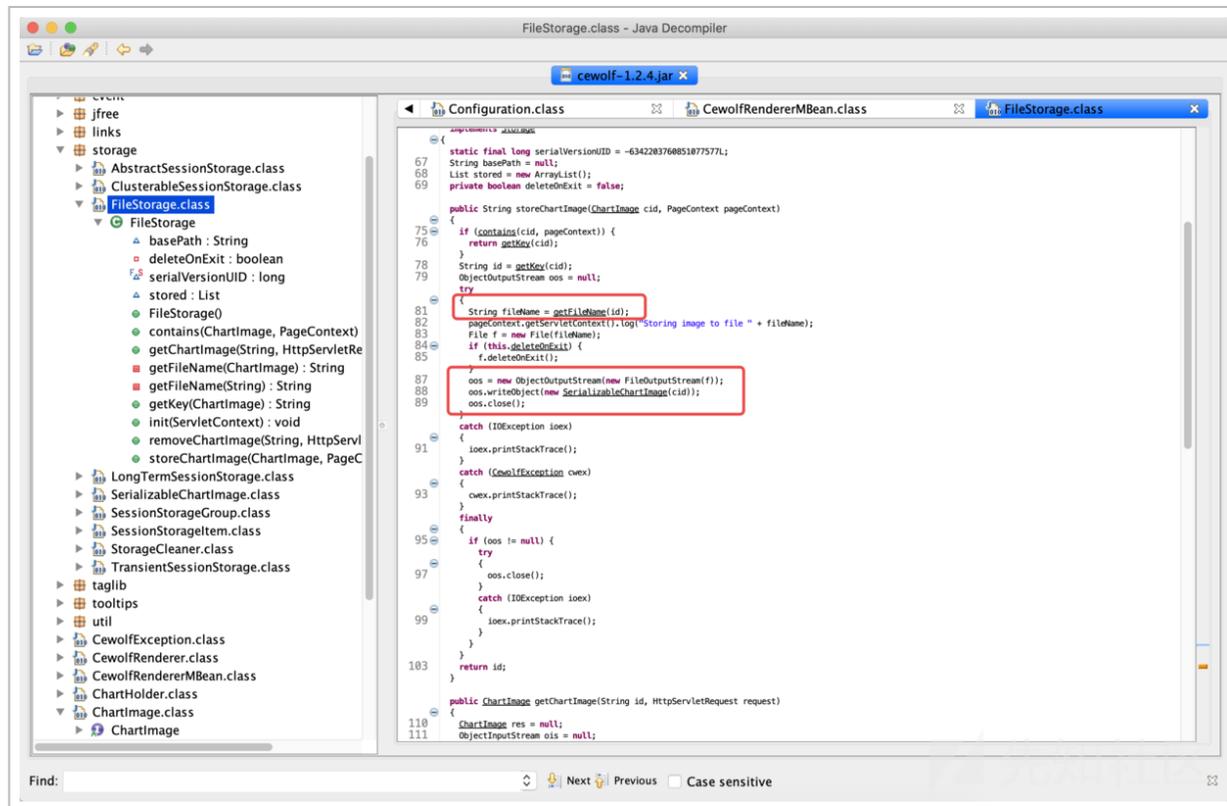
(<https://xzfile.aliyuncs.com/media/upload/picture/20200315111829-a474eac0-666b-1.png>)

imgKey 将会被 Storage 类中的 getChartImage 方法调用

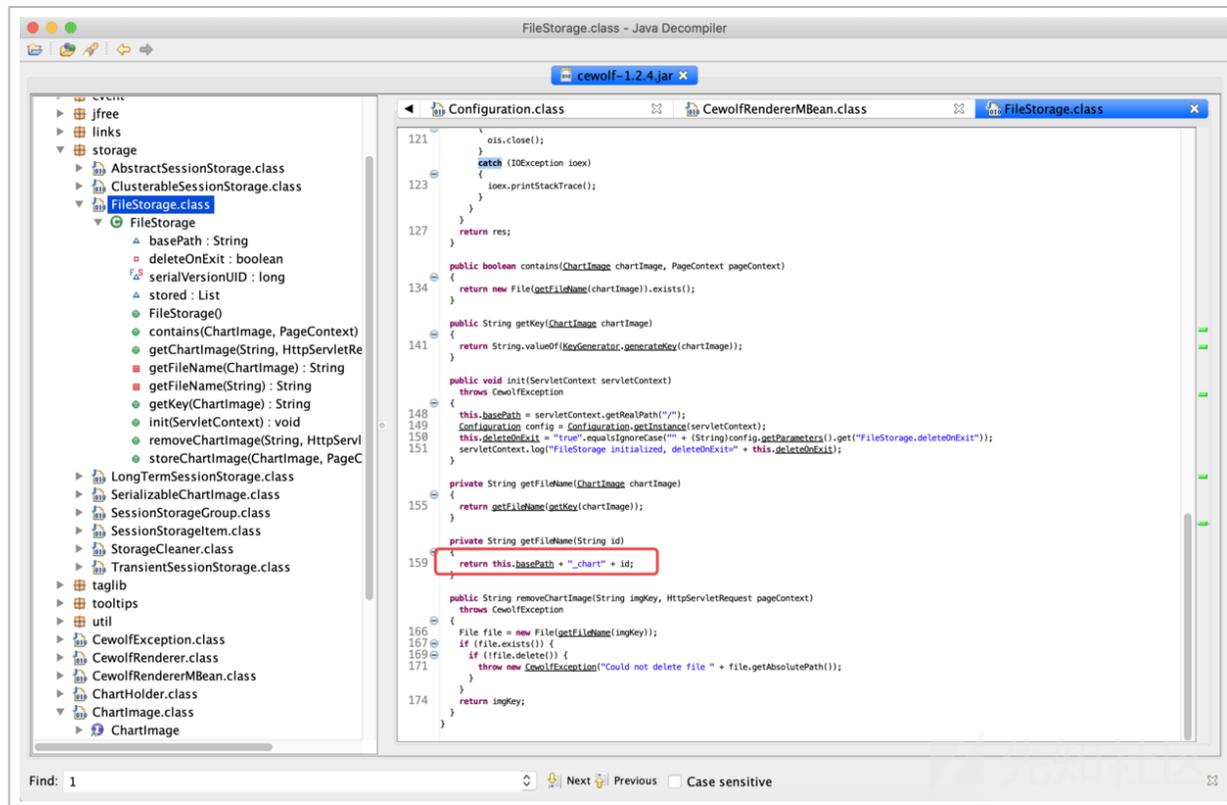
```
162     width = Integer.parseInt(request.getParameter("width"));
163     }
164     if (request.getParameter("height") != null) {
165         height = Integer.parseInt(request.getParameter("height"));
166     }
167     if (!renderingEnabled)
168     {
169         renderNotEnabled(response, 400, 50);
170         return;
171     }
172     if ((width > this.config.getMaxImageWidth()) || (height > this.config.getMaxImageHeight()))
173     {
174         renderImageTooLarge(response, 400, 50);
175         return;
176     }
177     String imgKey = request.getParameter("img");
178     if (imgKey == null)
179     {
180         logAndRenderException(new ServletException("no 'img' parameter provided for Cewolf servlet."), response, width, height);
181         return;
182     }
183     Storage storage = this.config.getStorage();
184     ChartImage chartImage = storage.getChartImage(imgKey, request);
185     if (chartImage == null)
186     {
187         renderImageExpired(response, 400, 50);
188         return;
189     }
190     requestCount.incrementAndGet();
191     try
192     {
193         long start = System.currentTimeMillis();
194         int size = chartImage.getSize();
195         response.setContentType(chartImage.getContentType());
196         response.setContentLength(size);
197         response.setBufferSize(size);
198         response.setStatus(200);
199         response.getOutputStream().write(chartImage.getBytes());
200         long last = System.currentTimeMillis() - start;
201         if (debugged) {
202             log("creation time for chart " + imgKey + ": " + last + "ms.");
203         }
204         return;
205     }
206     catch (Throwable t)
207     {
208         logAndRenderException(t, response, width, height);
209     }
210     finally
211     {
212         if (removeAfterRendering) {
213             try
214             {
215                 storage.removeAfterRendering(imgKey);
216             }
217             catch (Exception e)
218             {
219                 logAndRenderException(e, response, width, height);
220             }
221         }
222     }
223 }
```

(<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` .

```
<servlet>
  <servlet-name>AppCatalogServiceServlet</servlet-name>
  <servlet-class>com.me.mdm.agent.servlets.appcatalog.AppCatalogServiceServlet</servlet-class>
</servlet>

<servlet>
  <servlet-name>MDMLogUploaderServlet</servlet-name>
  <servlet-class>com.me.mdm.onpremise.webclient.log.MDMLogUploaderServlet</servlet-class>
</servlet>

<servlet>
  <servlet-name>MDMFileUploaderServlet</servlet-name>
  <servlet-class>com.me.mdm.agent.servlets.file.MDMFileUploaderServlet</servlet-class>
</servlet>

<!-- Added For Forwarding Server -->
  <servlet>
    <servlet-name>ForwardingServerStatusServlet</servlet-name>
    <servlet-class>com.me.dconpremise.webclient.servlet.ForwardingServerStatusServlet</servlet-class>
  </servlet>
```

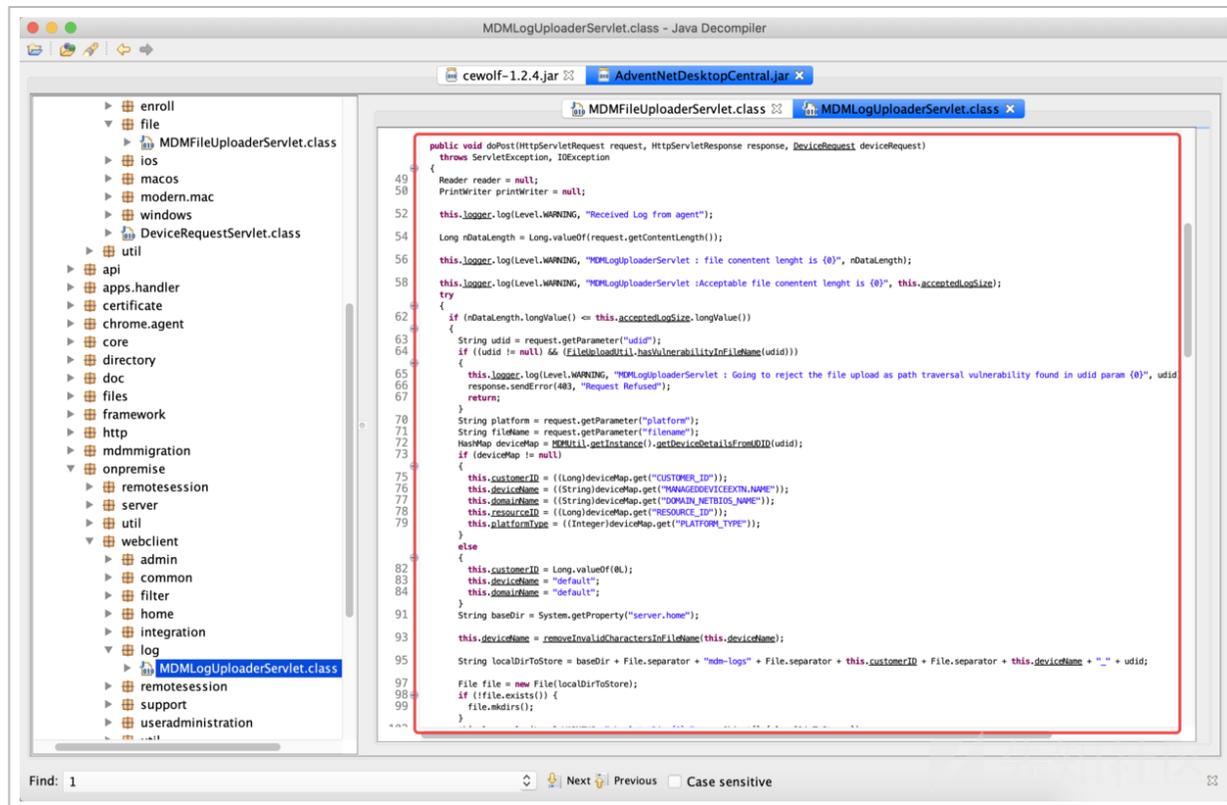
(<https://xzfile.aliyuncs.com/media/upload/picture/20200315112057-fcbbfec6-666b-1.png>)

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

```
public void doPost(HttpServletRequest request, HttpServletResponse response, DeviceRequest deviceRequest)
    throws ServletException, IOException
{
    Reader reader = null;
    PrintWriter printWriter = null;
    this.logger.log(Level.WARNING, "Received Log from agent");
    Long nDataLength = Long.valueOf(request.getContentLength());
    this.logger.log(Level.WARNING, "MDMLogUploaderServlet : file content length is {0}", nDataLength);
    this.logger.log(Level.WARNING, "MDMLogUploaderServlet :Acceptable file content length is {0}", this.acceptedLogSize);
    try
    {
        if (nDataLength.longValue() <= this.acceptedLogSize.longValue())
        {
            String uuid = request.getParameter("uuid");
            if ((uuid != null) && (FileUploadUtil.hasVulnerabilityOfFileName(uuid)))
            {
                this.logger.log(Level.WARNING, "MDMLogUploaderServlet : Going to reject the file upload as path traversal vulnerability found in uuid param {0}", uuid);
                response.sendError(403, "Request Refused");
                return;
            }
            String platform = request.getParameter("platform");
            String fileName = request.getParameter("filename");
            HashMap deviceMap = MDMUtil.getInstance().getDeviceDetailsFromID(uuid);
            if (deviceMap != null)
            {
                this.customerID = ((Long)deviceMap.get("CUSTOMER_ID"));
                this.deviceName = ((String)deviceMap.get("MANAGEDDEVICEID.NAME"));
                this.domainName = ((String)deviceMap.get("DOMAIN_NETBIOS_NAME"));
                this.resourceID = ((Long)deviceMap.get("RESOURCE_ID"));
                this.platformType = ((Integer)deviceMap.get("PLATFORM_TYPE"));
            }
            else
            {
                this.customerID = Long.valueOf(0L);
                this.deviceName = "default";
                this.domainName = "default";
            }
            String baseDir = System.getProperty("server.home");
            this.deviceName = removeInvalidCharactersOfFileName(this.deviceName);
            String localDirToStore = baseDir + File.separator + "mdm-logs" + File.separator + this.customerID + File.separator + this.deviceName + "_" + uuid;
            File file = new File(localDirToStore);
            if (file.exists()) {
                file.mkdirs();
            }
        }
    }
}
```

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

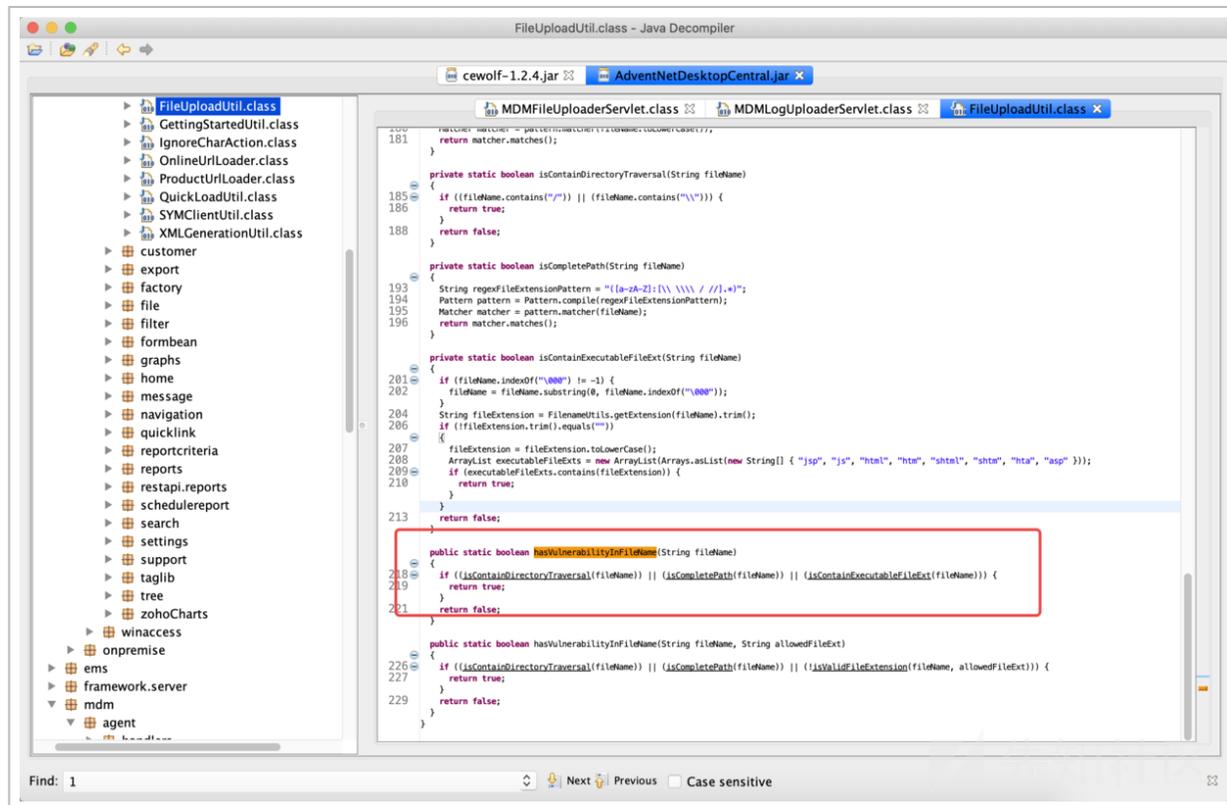
在该类中存在 `Post` 方法, 可见 `udid` 和 `filename` 为可控点, 我们完全可以自己传值控制. 当然 `udid` 其实也做了三种安全检查, 但并不影响目录穿越的产生.



```
MDMLogUploaderServlet.class - Java Decompiler
cewolf-1.2.4.jar
AdventNetDesktopCentral.jar
MDMFileUploaderServlet.class
MDMLogUploaderServlet.class

public void doPost(HttpServletRequest request, HttpServletResponse response, DeviceRequest deviceRequest)
    throws ServletException, IOException
{
    49 Reader reader = null;
    50 PrintWriter printWriter = null;
    52 this.logger.log(Level.WARNING, "Received Log from agent");
    54 Long nDataLength = Long.valueOf(request.getContentLength());
    56 this.logger.log(Level.WARNING, "MDMLogUploaderServlet : file content length is {0}", nDataLength);
    58 this.logger.log(Level.WARNING, "MDMLogUploaderServlet :Acceptable file content length is {0}", this.acceptedLogSize);
    try
    {
    62 if (nDataLength.longValue() <= this.acceptedLogSize.longValue())
    {
    63 String udid = request.getParameter("udid");
    64 if ((udid != null) && (Files.isWritable(Path.of(udid))))
    {
    65 this.logger.log(Level.WARNING, "MDMLogUploaderServlet : Going to reject the file upload as path traversal vulnerability found in udid param {0}", udid);
    66 response.sendError(403, "Request Refused");
    67 }
    70 String platform = request.getParameter("platform");
    71 String fileName = request.getParameter("filename");
    72 HashMap deviceMap = MDMUtil.getInstance().getDeviceDetailsFromID(udid);
    73 if (deviceMap != null)
    {
    75 this.customerID = ((Long)deviceMap.get("CUSTOMER_ID"));
    76 this.deviceName = ((String)deviceMap.get("MANAGED_DEVICE_CN_NAME"));
    77 this.domainName = ((String)deviceMap.get("DOMAIN_NETBIOS_NAME"));
    78 this.resourceID = ((Long)deviceMap.get("RESOURCE_ID"));
    79 this.platformType = ((Integer)deviceMap.get("PLATFORM_TYPE"));
    }
    82 else
    {
    83 this.customerID = Long.valueOf(0L);
    84 this.deviceName = "default";
    85 this.domainName = "default";
    }
    91 String baseDir = System.getProperty("server.home");
    93 this.deviceName = removeInvalidCharactersInFileName(this.deviceName);
    95 String localDirToStore = baseDir + File.separator + "mdm-logs" + File.separator + this.customerID + File.separator + this.deviceName + "_" + udid;
    97 File file = new File(localDirToStore);
    98 if (file.exists()) {
    99 file.mkdirs();
    }
    }
}
```

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



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

```
FileUploadUtil.class - Java Decompiler
cewolf-1.2.4.jar
AdventNetDesktopCentral.jar
MDMFileUploaderServlet.class
MDMLogUploaderServlet.class
FileUploadUtil.class

FileUploadUtil.class
  GettingStartedUtil.class
  IgnoreCharAction.class
  OnlineUrlLoader.class
  ProductUrlLoader.class
  QuickLoadUtil.class
  SYMClientUtil.class
  XMLGenerationUtil.class
  customer
  export
  factory
  file
  filter
  formbean
  graphs
  home
  message
  navigation
  quicklink
  reportcriteria
  reports
  restapi.reports
  schedulerreport
  search
  settings
  support
  taglib
  tree
  zohoCharts
  winaccess
  onpremise
  ems
  framework.server
  mdm
  agent

160     }
161     folder.delete();
162     this.out.log(Level.INFO, "Deleted : " + fileFolder);
163 }
164 catch (Exception e)
165 {
166     returnType = false;
167     this.out.log(Level.WARNING, "Exception occured while deleting folder.", e);
168 }
169 return returnType;
170 }

171 public static boolean isValidFileExtension(String fileName, String fileExtPattern)
172 {
173     if (fileName.indexOf(".") != -1) {
174         fileName = fileName.substring(0, fileName.indexOf("."));
175     }
176     fileExtPattern = fileExtPattern.replaceAll("\\s+", "").toLowerCase();
177     String regexFileExtensionPattern = "(\\.|\\.|(?!))" + fileExtPattern + "$";
178     Pattern pattern = Pattern.compile(regexFileExtensionPattern);
179     Matcher matcher = pattern.matcher(fileName.toLowerCase());
180     return matcher.matches();
181 }

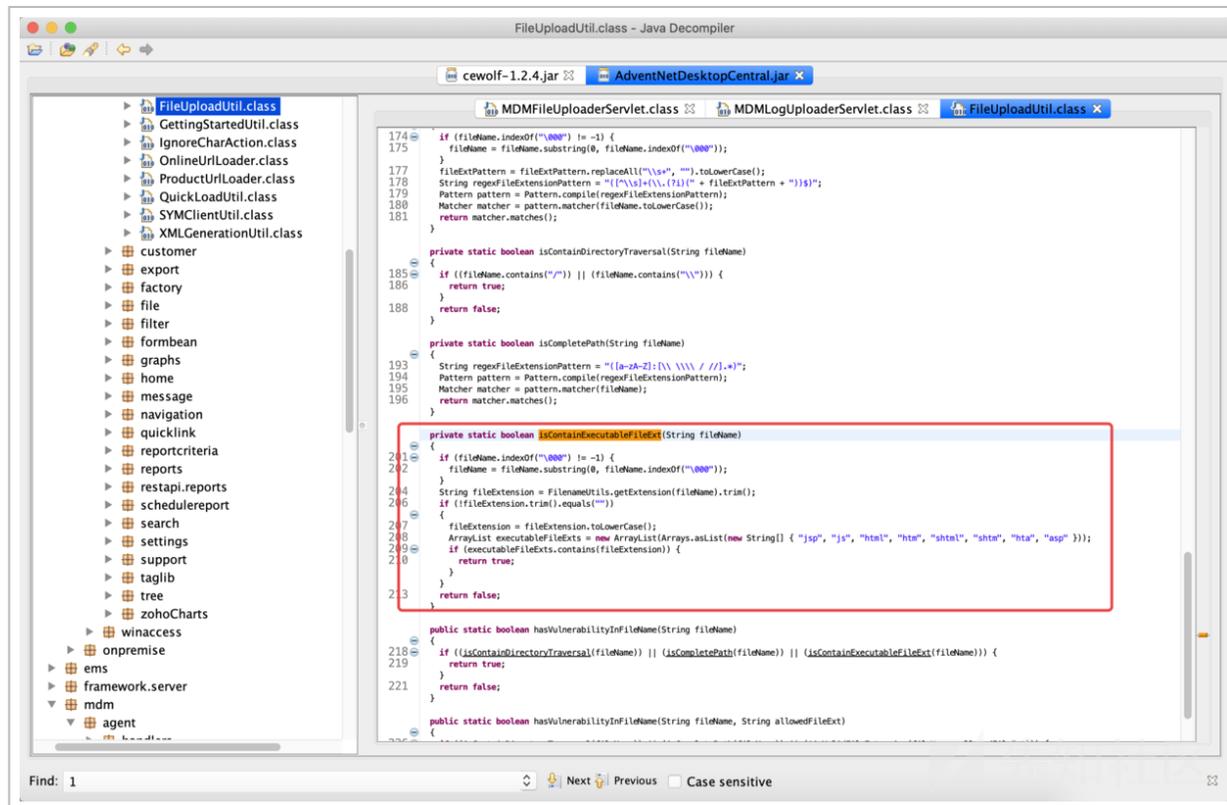
182 private static boolean isContainDirectoryTraversa(String fileName)
183 {
184     if (fileName.contains("/") || fileName.contains("\\")) {
185         return true;
186     }
187     return false;
188 }

189 private static boolean isCompletePath(String fileName)
190 {
191     String regexFileExtensionPattern = "[a-zA-Z]\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\";
192     Pattern pattern = Pattern.compile(regexFileExtensionPattern);
193     Matcher matcher = pattern.matcher(fileName);
194     return matcher.matches();
195 }

196 private static boolean isContainExecutableFileExt(String fileName)
197 {
198     if (fileName.indexOf(".") != -1) {
199         fileName = fileName.substring(0, fileName.indexOf("."));
200     }
201     String fileExtension = FileUtil.getFileExtension(fileName).trim();
202     if (fileExtension.trim().equals("")) {
203         return false;
204     }
205     fileExtension = fileExtension.toLowerCase();
206     ArrayList executableFileExts = new ArrayList(Arrays.asList(new String[] { ".jsp", ".js", ".html", ".htm", ".shtml", ".shtm", ".hta", ".asp" }));
207     if (executableFileExts.contains(fileExtension)) {
208         return true;
209     }
210     return false;
211 }

Find: 1
Next Previous Case sensitive
```


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



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

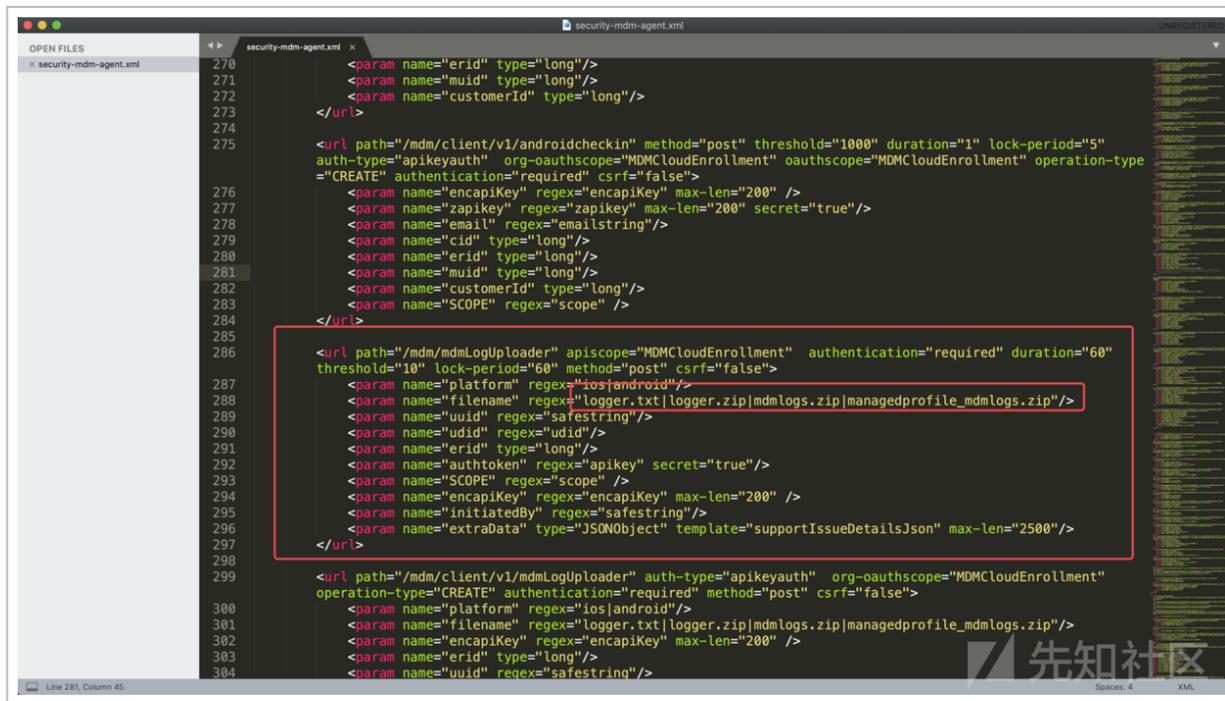
而 `filename` 被做了后缀名限制, 只允许为 `log|txt|zip|7z` 其中一种

```
MDMLogUploaderServlet.class - Java Decompiler
cewolf-1.2.4.jar
AdventNetDesktopCentral.jar
MDMFileUploaderServlet.class
MDMLogUploaderServlet.class
FileUploadUtil.class

91 }
92 String baseDir = System.getProperty("server.home");
93 this.deviceName = removeInvalidCharactersInFilename(this.deviceName);
95 String localDirToStore = baseDir + File.separator + "mdm-logs" + File.separator + this.customerID + File.separator + this.deviceName + "_" + uuid;
97 File file = new File(localDirToStore);
98 if (file.exists()) {
99     file.mkdirs();
102 }
103 this.logger.log(Level.WARNING, "absolute Dir () ", new Object[] { localDirToStore });
108 filename = filename.toLowerCase();
110 if ((filename != null) && !FileUploadUtil.hasVulnerabilityInFilename(filename, "log|txt|zip|7z"))
111 {
112     this.logger.log(Level.WARNING, "MDMLogUploaderServlet : Going to reject the file upload ()", filename);
113     response.sendError(403, "Request Refused");
114     return;
116 }
117 String absoluteFileName = localDirToStore + File.separator + filename;
118 this.logger.log(Level.WARNING, "absolute File Name () ", new Object[] { filename });
120 InputStream fileStreamData = null;
121 try
122 {
123     fileStreamData = validateRequestStreamWithFile(request.getInputStream(), filename);
124 }
125 catch (SQLException ex)
126 {
127     if (ex.getErrorCode() == 140001)
128     {
129         this.logger.log(Level.WARNING, "MDMLogUploaderServlet : Content type detected is not application/zip - Terminating log upload request with error code");
130         response.sendError(403, "Request Refused");
131         return;
132     }
133 }
134 InputStream in = null;
135 FileOutputStream fout = null;
136 try
137 {
138     in = fileStreamData;
139     fout = new FileOutputStream(absoluteFileName);
140     byte[] bytes = new byte[4096];
141     int i;
142     while ((i = in.read(bytes)) != -1) {
143         fout.write(bytes, 0, i);
144     }
145     fout.flush();
146 }
```

(<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



```
270 <param name="erid" type="long"/>
271 <param name="muid" type="long"/>
272 <param name="customerId" type="long"/>
273 </url>
274
275 <url path="/mdm/client/v1/androidcheckin" method="post" threshold="1000" duration="1" lock-period="5"
276 auth-type="apikeyauth" org-oauthscope="MDMCloudEnrollment" oauthscope="MDMCloudEnrollment" operation-type
277 ="CREATE" authentication="required" csrf="false">
278 <param name="encapiKey" regex="encapiKey" max-len="200" />
279 <param name="zapikey" regex="zapikey" max-len="200" secret="true"/>
280 <param name="email" regex="emailstring"/>
281 <param name="cid" type="long"/>
282 <param name="erid" type="long"/>
283 <param name="muid" type="long"/>
284 <param name="customerId" type="long"/>
285 <param name="SCOPE" regex="scope" />
286 </url>
287
288 <url path="/mdm/mdmLogUploader" apiscope="MDMCloudEnrollment" authentication="required" duration="60"
289 threshold="10" lock-period="60" method="post" csrf="false">
290 <param name="platform" regex="ios|android"/>
291 <param name="filename" regex="logger.txt|logger.zip|mdmlogs.zip|managedprofile_mdmlogs.zip"/>
292 <param name="uuid" regex="safestring"/>
293 <param name="udid" regex="udid"/>
294 <param name="erid" type="long"/>
295 <param name="authToken" regex="apikey" secret="true"/>
296 <param name="SCOPE" regex="scope" />
297 <param name="encapiKey" regex="encapiKey" max-len="200" />
298 <param name="initiatedBy" regex="safestring"/>
299 <param name="extraData" type="JSONObject" template="supportIssueDetailsJson" max-len="2500"/>
300 </url>
301
302 <url path="/mdm/client/v1/mdmLogUploader" auth-type="apikeyauth" org-oauthscope="MDMCloudEnrollment"
303 operation-type="CREATE" authentication="required" method="post" csrf="false">
304 <param name="platform" regex="ios|android"/>
305 <param name="filename" regex="logger.txt|logger.zip|mdmlogs.zip|managedprofile_mdmlogs.zip"/>
306 <param name="encapiKey" regex="encapiKey" max-len="200" />
307 <param name="erid" type="long"/>
308 <param name="udid" regex="safestring"/>
```

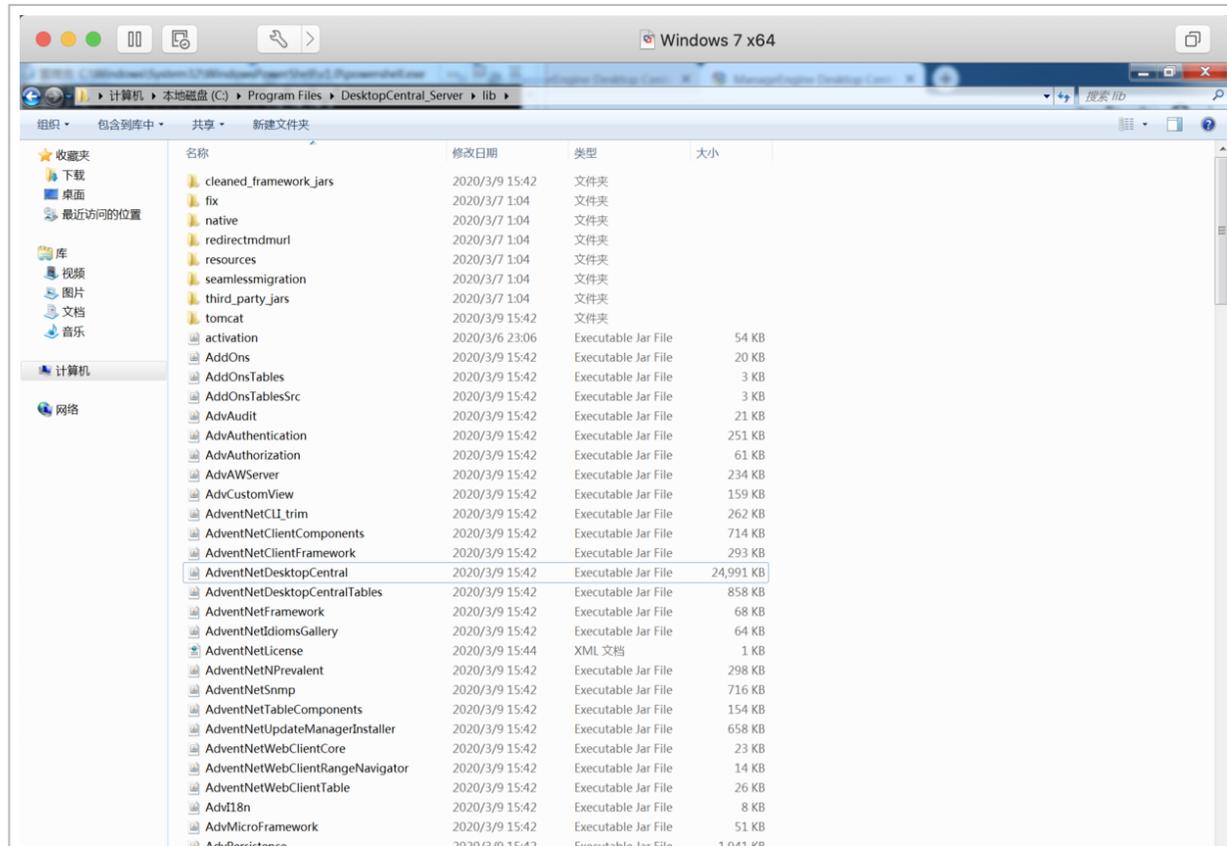
(<https://xzfile.aliyuncs.com/media/upload/picture/20200315112529-9f0c40dc-666c-1.png>)

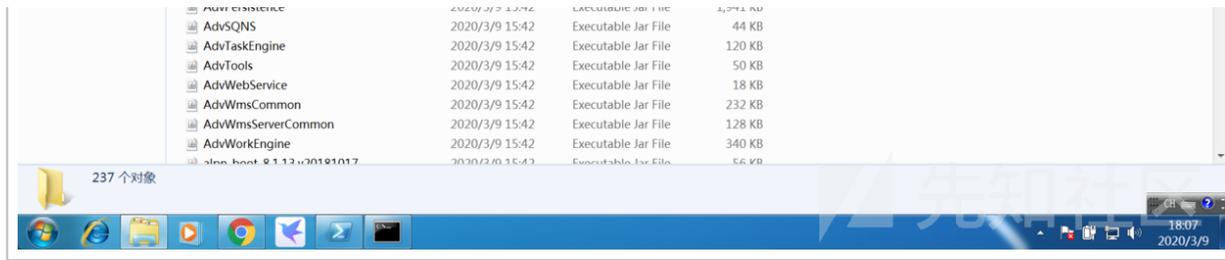
1.png)

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

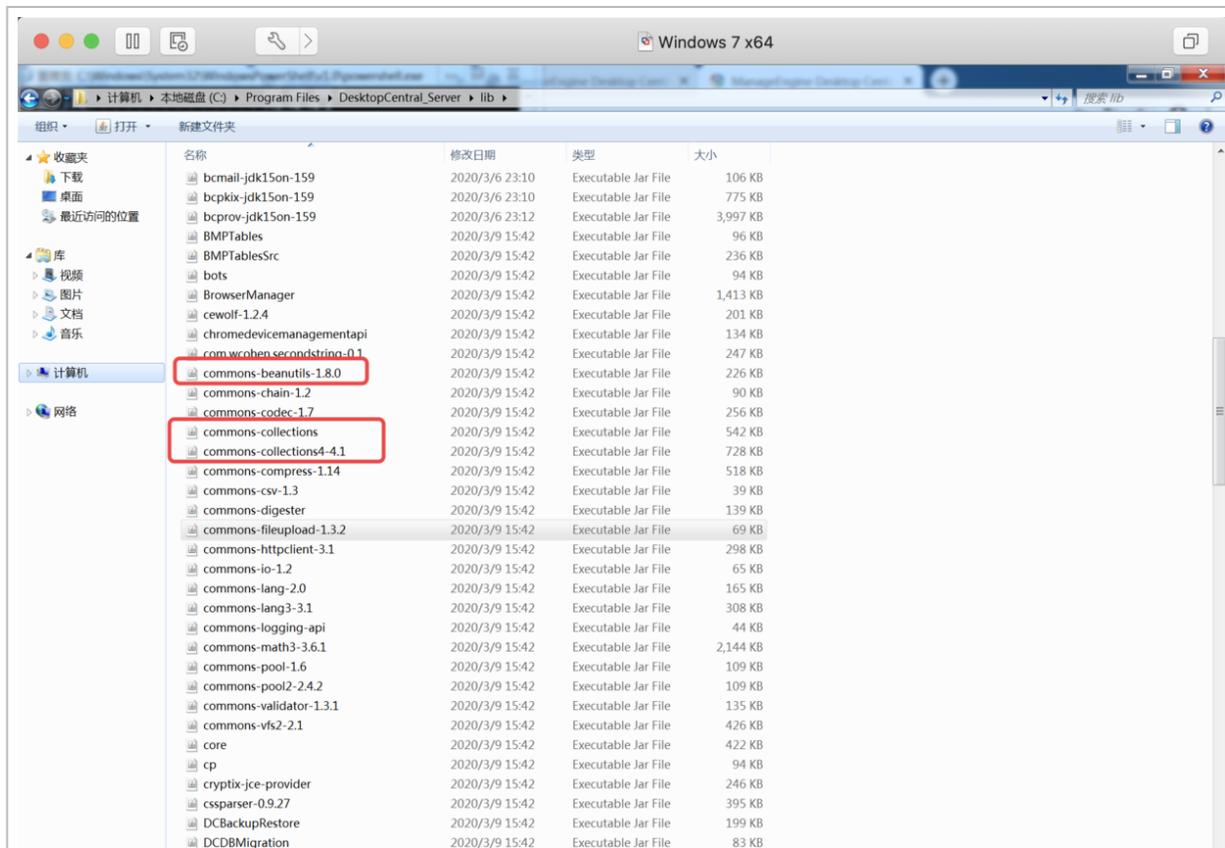
我们进入 `DesktopCentral_Server\lib` 目录, 寻找有漏洞的 jar.

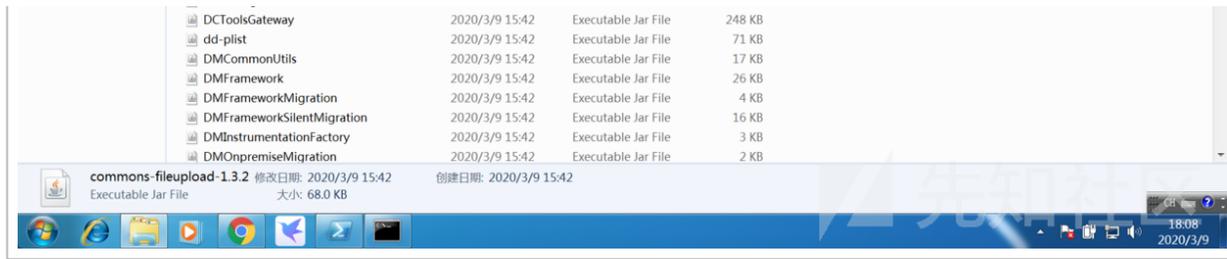




(<https://xzfile.aliyuncs.com/media/upload/picture/20200315112637-c79fc924-666c-1.png>)

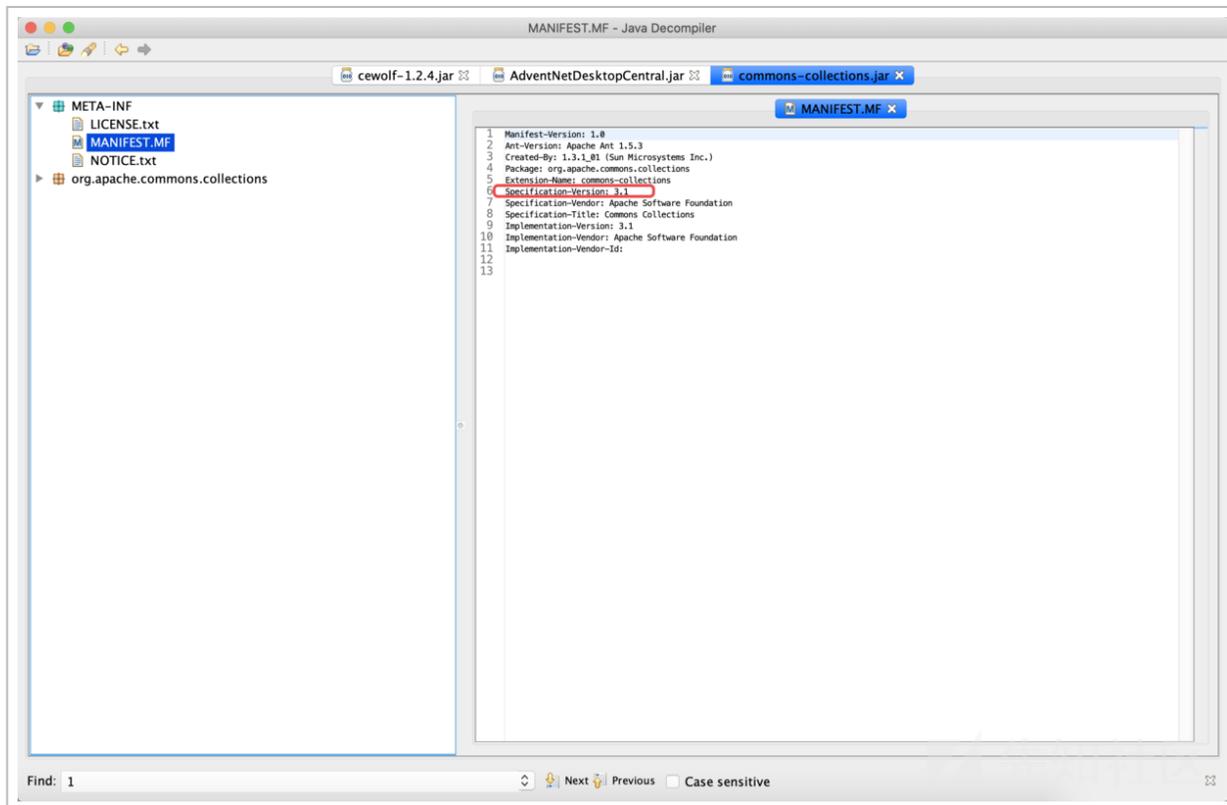
可见我们发现了 `commons-collections.jar` 、 `commons-beanutils-1.8.0.jar`





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

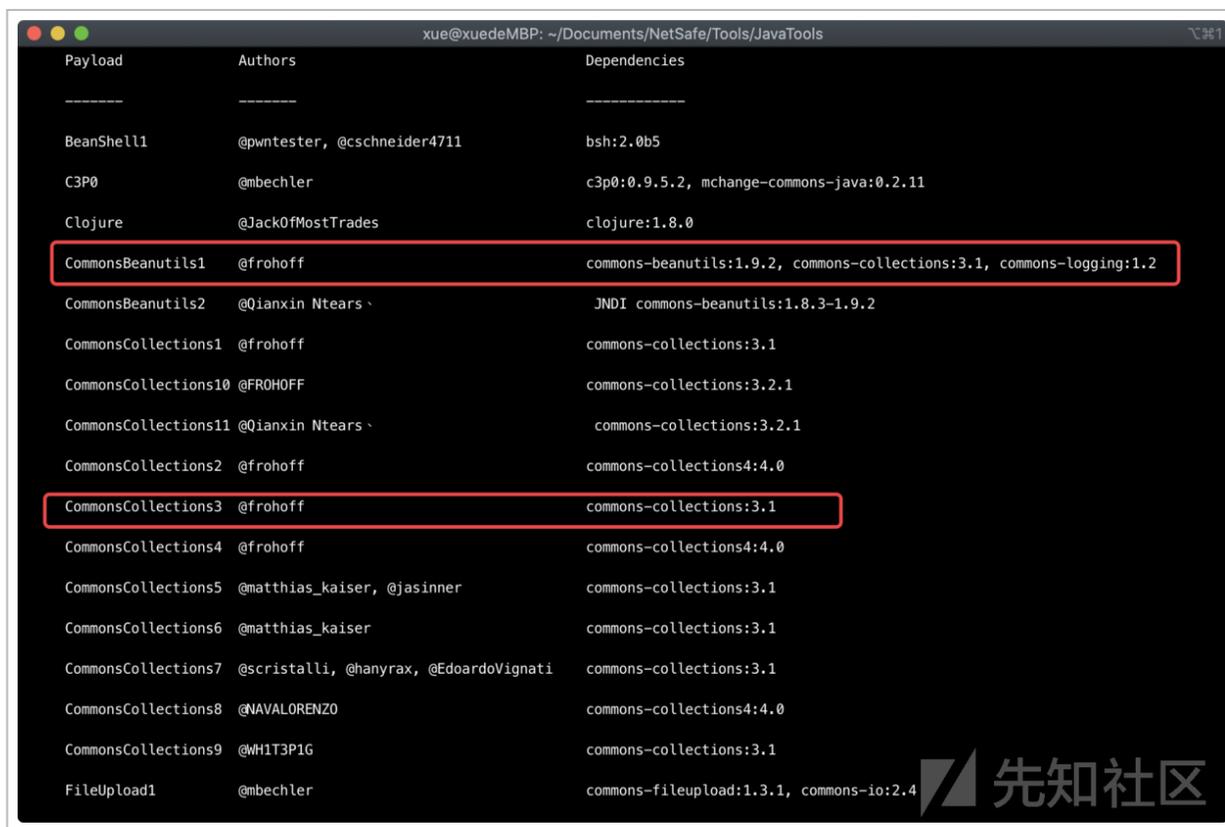
commons-collections.jar 不确定具体版本, 我们查看一下版本



(<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 gadgets. 当然还有 jdk 的两条 gadgets, `JDK7U21` 和 `JRE8U20` .

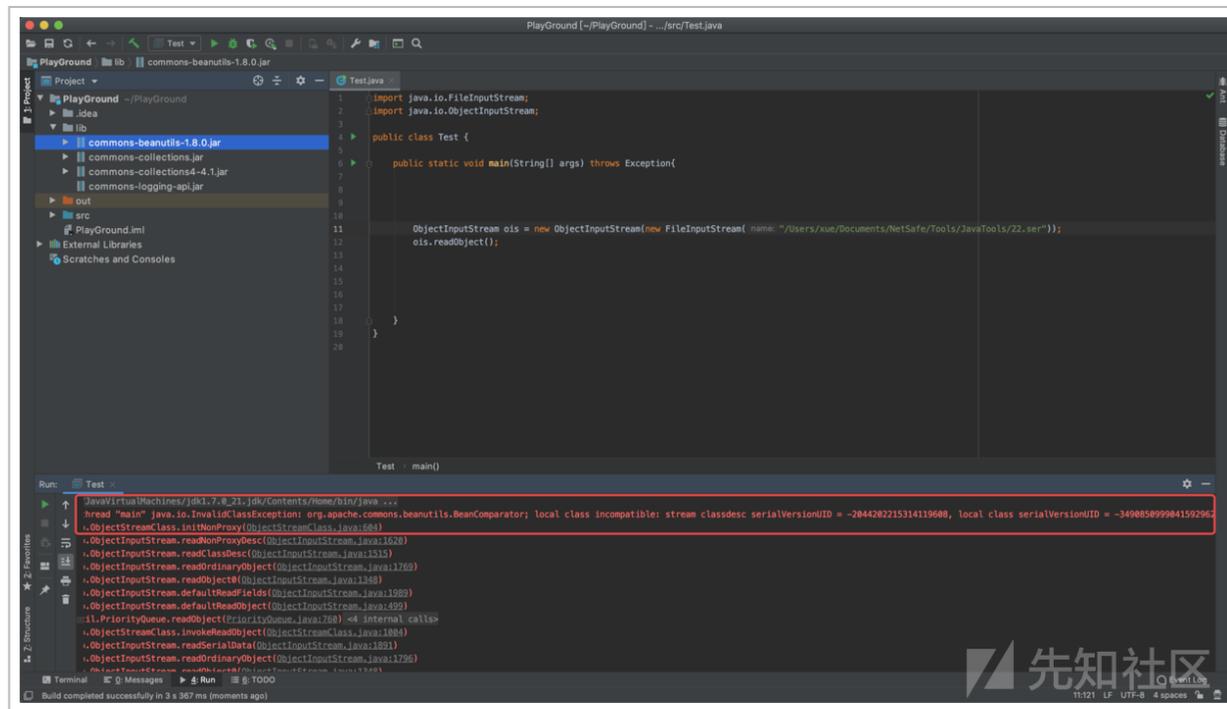
我们请出反序列化文件构建神器—— `ysoserial` .



```
xue@xuedeMBP: ~/Documents/NetSafe/Tools/JavaTools
Payload      Authors      Dependencies
-----
BeanShell1   @pwntester, @cschneider4711  bsh:2.0b5
C3P0         @mbechler    c3p0:0.9.5.2, mchange-commons-java:0.2.11
Clojure      @JackOfMostTrades  clojure:1.8.0
CommonsBeanutils1 @frohoff     commons-beanutils:1.9.2, commons-collections:3.1, commons-logging:1.2
CommonsBeanutils2 @Qianxin Ntears `  JNDI commons-beanutils:1.8.3-1.9.2
CommonsCollections1 @frohoff     commons-collections:3.1
CommonsCollections10 @FR0HOFF     commons-collections:3.2.1
CommonsCollections11 @Qianxin Ntears `  commons-collections:3.2.1
CommonsCollections2 @frohoff     commons-collections:4.4.0
CommonsCollections3 @frohoff     commons-collections:3.1
CommonsCollections4 @frohoff     commons-collections:4.4.0
CommonsCollections5 @matthias_kaiser, @jasinner  commons-collections:3.1
CommonsCollections6 @matthias_kaiser  commons-collections:3.1
CommonsCollections7 @scristalli, @hanyrax, @EdoardoVignati  commons-collections:3.1
CommonsCollections8 @NAVALORENZO     commons-collections:4.4.0
CommonsCollections9 @WH1T3P1G       commons-collections:3.1
FileUpload1   @mbechler    commons-fileupload:1.3.1, commons-io:2.4
```

(<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>)

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

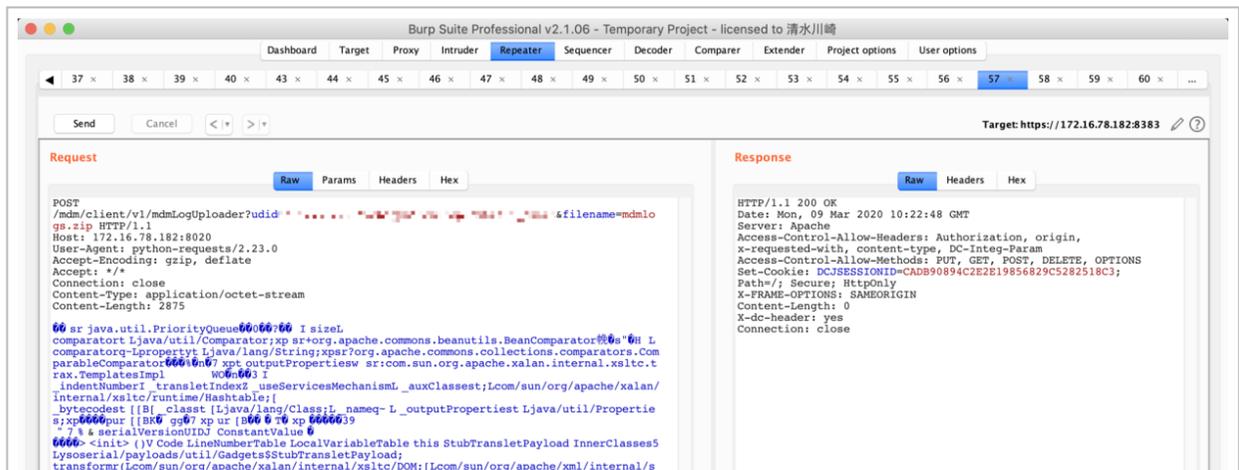
```
<dependency>
  <groupId>commons-collections</groupId>
  <artifactId>commons-collections</artifactId>
  <version>3.1</version>
</dependency>
<dependency>
  <groupId>org.beanshell</groupId>
  <artifactId>bsh</artifactId>
  <version>2.0b5</version>
</dependency>
<dependency>
  <groupId>commons-beanutils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

```
</dependency>
<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>
  <version>4.1.4.RELEASE</version>
</dependency>
```

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

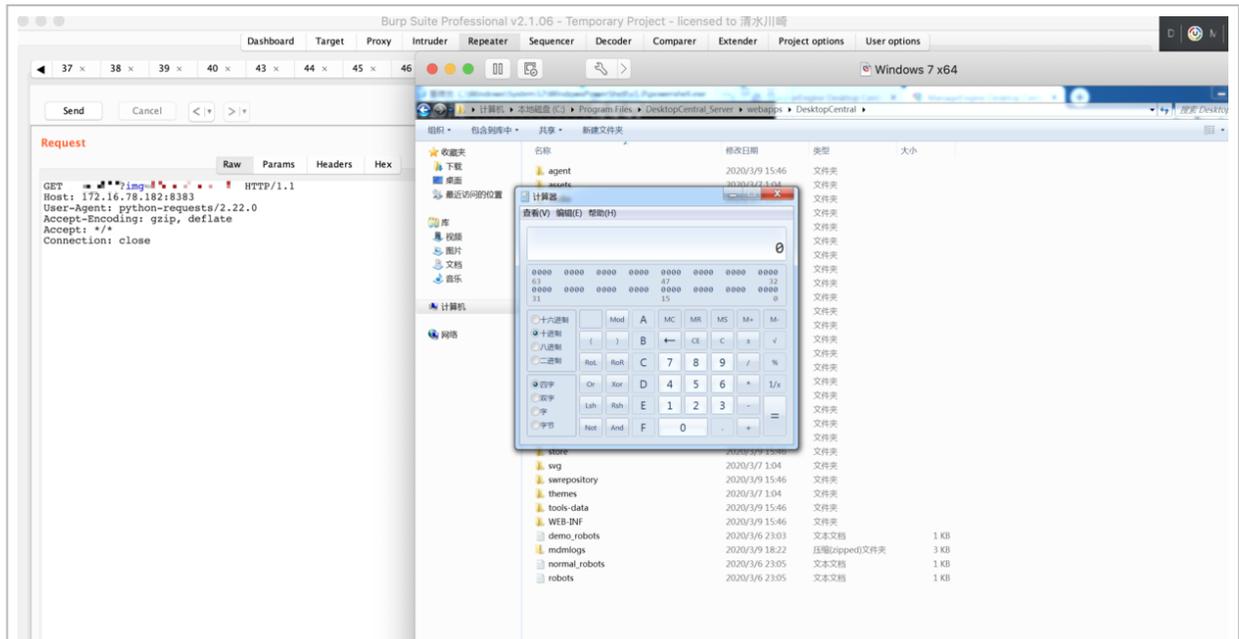
4. 效果演示

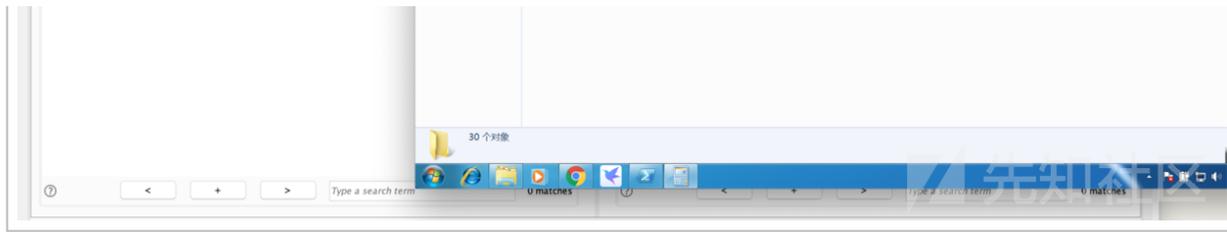
为了满足漏洞管理条例, 本文不直接放出 payload, 可自行参考文末的公开 payload





(<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>)