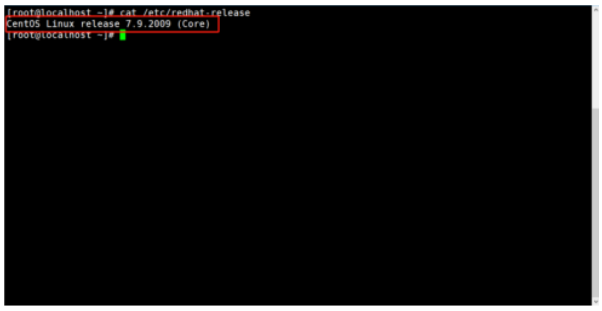
目录

OpenRASP 手册

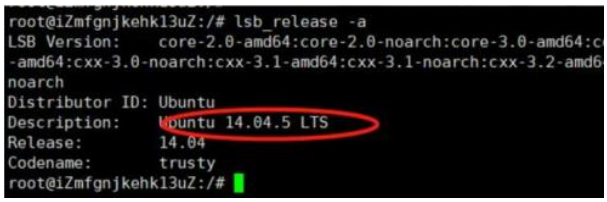
部署前先看

环境检查

1. （重要）安装前请确定系统环境是否符合如下条件：
   a) 中间件：
      i. Resin 3 ~ 4
      ii. Tomcat 8
      iii. WebLogic 12c
      iv. 宝兰德 9.5.x
      v. 东方通 7
   b) 服务器要求：
      i. Windows x64
      ii. MacOS 10.10.+
      iii. Linux：
         1. RHEL/CentOS 6~7，查看命令是：cat /etc/redhat-release

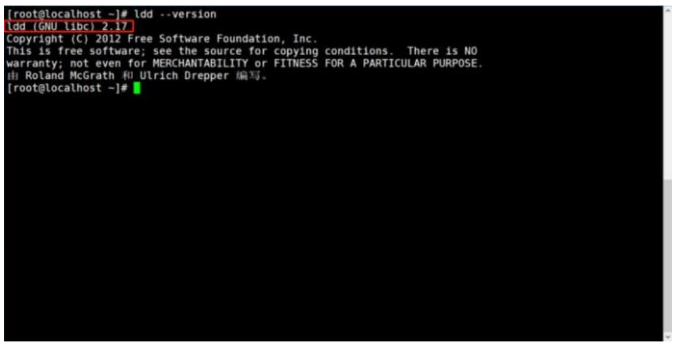


         2. Ubuntu 14 及更高版本，查看命令是：lsb_release -a

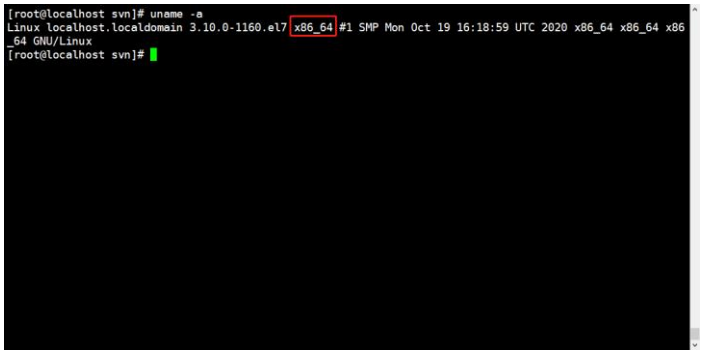


         3. Debian 6 及更高版本，查看命令是：cat /etc/debian_version

         4. 其他 glibc >= 2.12 的发行版，查看的命令是：ldd –version



5. **linux 服务器安装前请先确认操作系统的内核架构，目前已知 arm64 和 aarch64 架构不支持。查看命令是 uname -a**



6. **CentOS8 暂时不支持**

c) **运行中间件的 JDK 版本：**

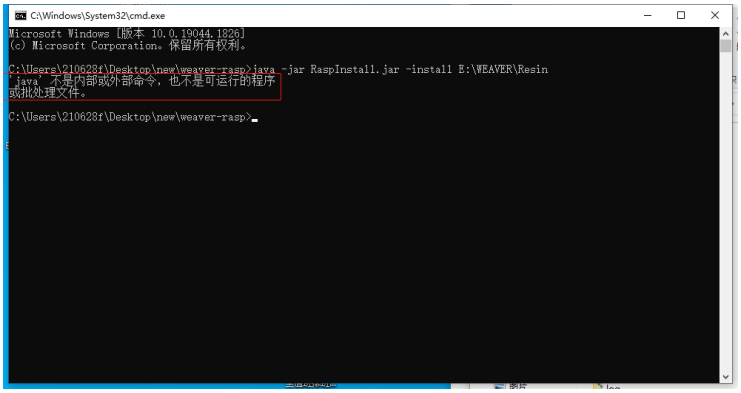
i. Oracle JDK 8

ii. Open JDK 8

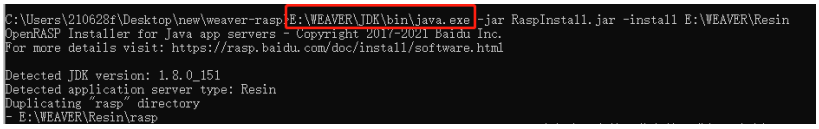
常见问题和解决方案

java 不是内部或外部命令

问题描述：在部署过程中，如果出现 **java 不是内部或外部命令**的错误

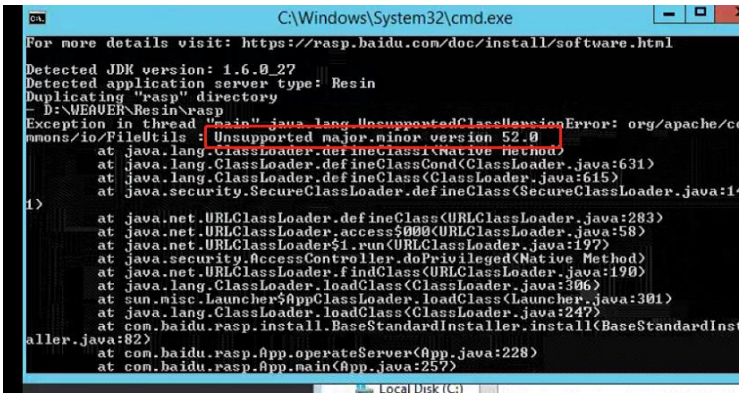


解决方案：将 java 命令替换为 weaver 路径下 java 的绝对路径+



Unsupported major.minor version 52.0

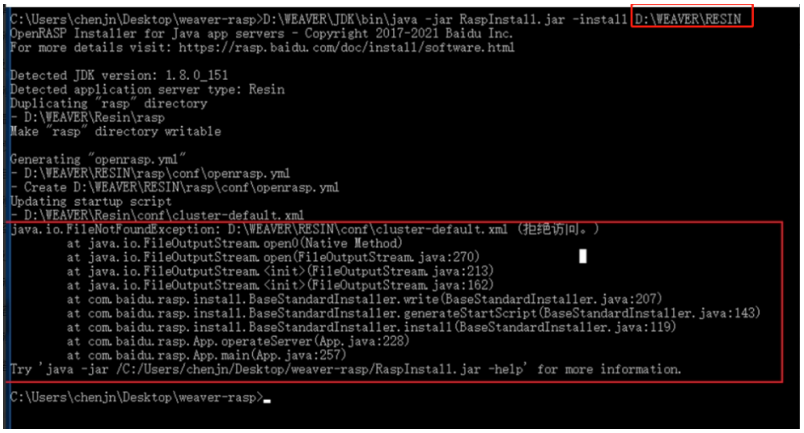
问题描述：如果在执行系统命令时使用了低版本的 java 命令，则会出现如下错误：



解决方法：参考 **Java 命令找不到**问题的解决方案。如果按照 **Java 命令找不到**问题的解决方案解决后仍然出现该问题，那么需要重新检查 OA 使用的 JDK 版本。

cluster-default.xml（拒绝访问）

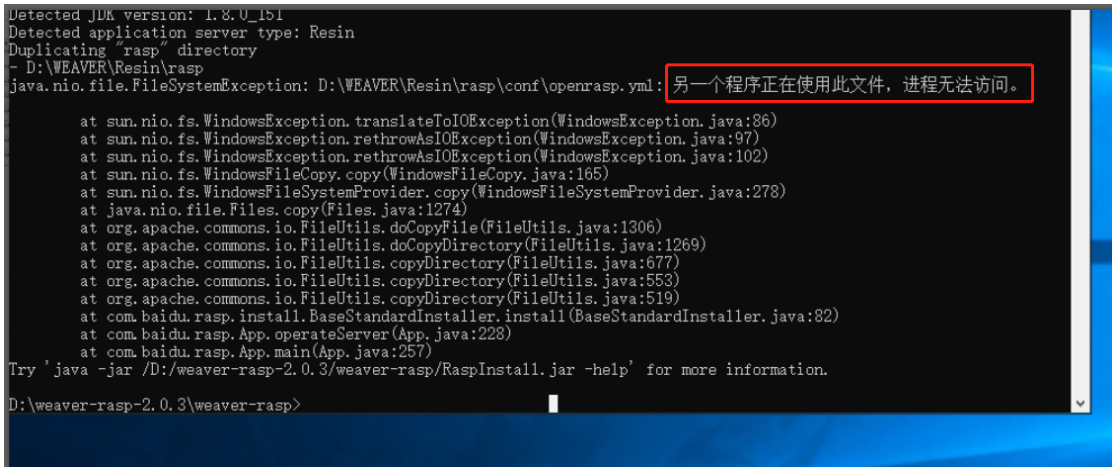
问题描述：在安装过程中可能会出现如下报错



解决方案：使用管理员身份运行该程序

另一个进程正在使用此文件

问题描述：安装过程中可能出现如下报错



解决方案：停止 OA 后再安装一次

部署方法

Resin3~4（安装包安装）

Linux

1. 执行如下命令，将 weaver-rasp.zip 解压至服务器任意目录

```
1. unzip weaver-rasp.zip -d ./
```

2. 进入解压后目录，将<解压位置>替换为真实路径

```
1. cd <解压位置>/weaver-rasp/
```

```
[root@localhost tomcat]# ls
apache-tomcat-8.5.77  weaver-rasp  weaver-rasp.zip
[root@localhost tomcat]# cd weaver-rasp
[root@localhost weaver-rasp]# ls
rasp  RaspInstall.jar
[root@localhost weaver-rasp]#
```

3. 执行安装命令，将命令中的<resin_root>替换为真实的 resin 绝对路径

```
1. java -jar RaspInstall.jar -install <resin_root>
```

```
apache-tomcat-8.5.77  weaver-rasp  weaver-rasp.zip
[root@localhost tomcat]# cd weaver-rasp
[root@localhost weaver-rasp]# ls
rasp  RaspInstall.jar
[root@localhost weaver-rasp]# java -jar RaspInstall.jar -install /usr/local/gs/resin/resin-4.0.
42/
OpenRASP Installer for Java app servers - Copyright 2017-2021 Baidu Inc.
For more details visit: https://rasp.baidu.com/doc/install/software.html

Detected JDK version: 1.8.0_321
Detected application server type: Resin
Duplicating "rasp" directory
- /usr/local/gs/resin/resin-4.0.42/rasp
Make "rasp" directory writable

Generating "openrasp.yml"
- /usr/local/gs/resin/resin-4.0.42/rasp/conf/openrasp.yml
- Backed up openrasp.yml to openrasp.yml.bak
Updating startup script
- /usr/local/gs/resin/resin-4.0.42/conf/cluster-default.xml

Installation completed without errors.
Please restart application server to take effect.
[root@localhost weaver-rasp]#
```

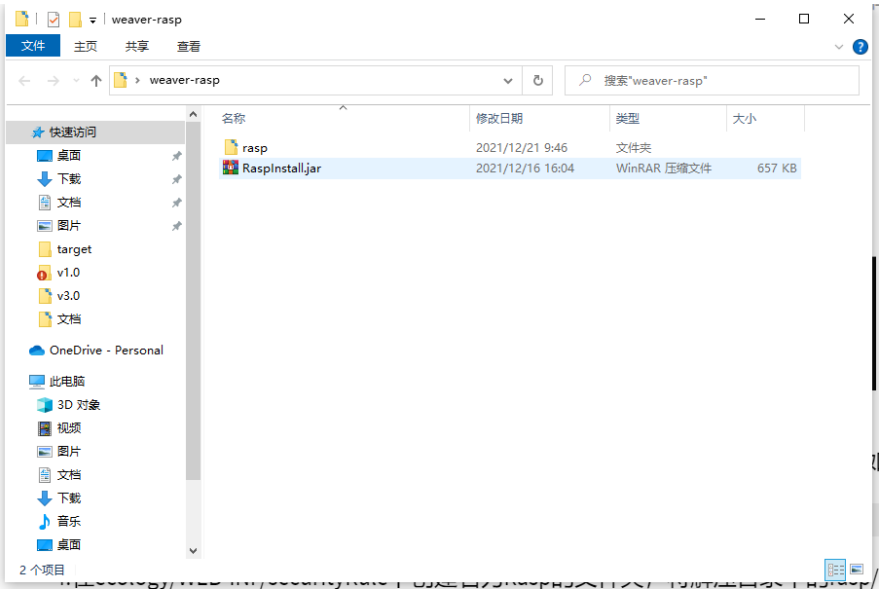
4. 在 ecology/WEB-INF/securityRule 下创建名为 Rasp 的文件夹，将解压目录中的 rasp/plugins/official.js 文件复制到该文件夹下

```
1. cd /home/weaver/ecology/WEB-INF/securityRule/
2. mkdir Rasp
3. cp /opt/rasp/weaver-rasp/rasp/plugins/official.js /home/weaver/ecology/WEB-INF/securityRule/Rasp
```

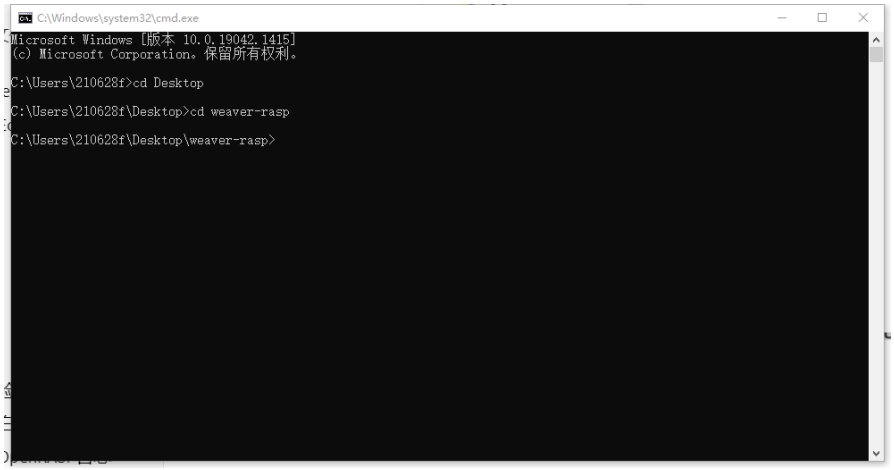
5. 安装步骤完毕，重启 OA 服务后 OpenRASP 生效

Windows

1. 将 weaver-rasp.zip 解压至服务器任意目录

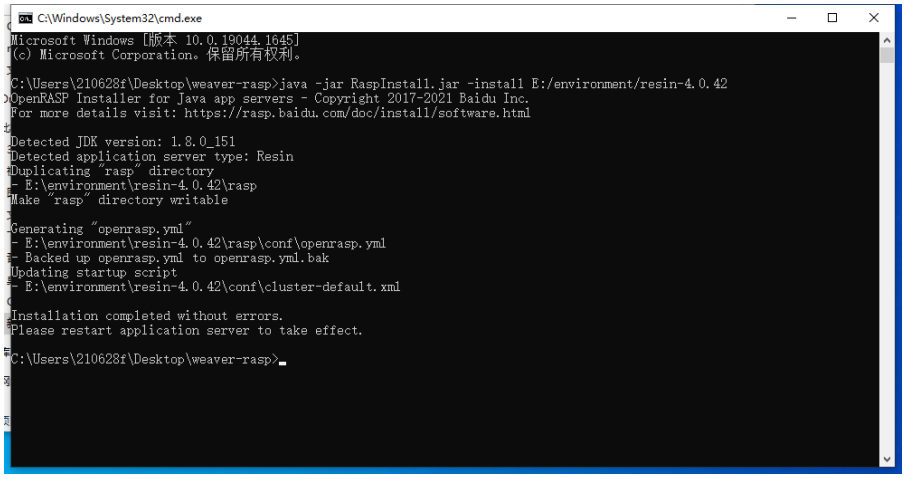


2. 通过 cmd，进入解压后的目录



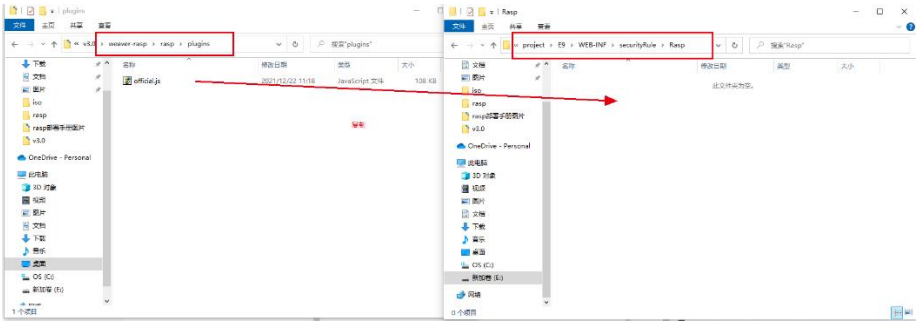
3. 在 cmd 中执行安装命令，将命令中的<resin_root>替换为真实的 resin 绝对路径，如：

1. java -jar RaspInstall.jar -install <resin_root>



4. 在 ecology/WEB-INF/securityRule 下创建名为 Rasp 的文件夹，将解压目录中

的 rasp/plugins/official.js 文件复制到该文件夹下



5. 安装步骤完毕，重启 OA 服务后 OpenRASP 生效

Tomcat（安装包安装）

Linux

1. 将 weaver-rasp.zip 解压至服务器任意目录

```
1. unzip weaver-rasp.zip -d ./
```

2. 进入解压后的目录，将<解压位置>替换为真实路径

```
1. cd <解压位置>/weaver-rasp/
```

```
[root@localhost tomcat]# cd weaver-rasp
[root@localhost weaver-rasp]# ls
rasp  RaspInstall.jar
[root@localhost weaver-rasp]#
```

3. 执行安装命令，将命令中的<tomcat_root>替换为真实的 tomcat 绝对路径，如：/home/weaver/tomcat/

1. java -jar RaspInstall.jar -install <tomcat_root>

```
[root@localhost tomcat]# cd weaver-rasp
[root@localhost weaver-rasp]# ls
rasp  RaspInstall.jar
[root@localhost weaver-rasp]# java -jar RaspInstall.jar -install /usr/local/gs/tomcat/apache-to
mcat-8.5.77/
OpenRASP Installer for Java app servers - Copyright 2017-2021 Baidu Inc.
For more details visit: https://rasp.baidu.com/doc/install/software.html

Detected JDK version: 1.8.0_321
Detected application server type: Tomcat
Duplicating "rasp" directory
- /usr/local/gs/tomcat/apache-tomcat-8.5.77/rasp
Make "rasp" directory writable

Generating "openrasp.yml"
- /usr/local/gs/tomcat/apache-tomcat-8.5.77/rasp/conf/openrasp.yml
- Backed up openrasp.yml to openrasp.yml.bak
Updating startup script
- /usr/local/gs/tomcat/apache-tomcat-8.5.77/bin/catalina.sh

Installation completed without errors.
Please restart application server to take effect.
[root@localhost weaver-rasp]#
```

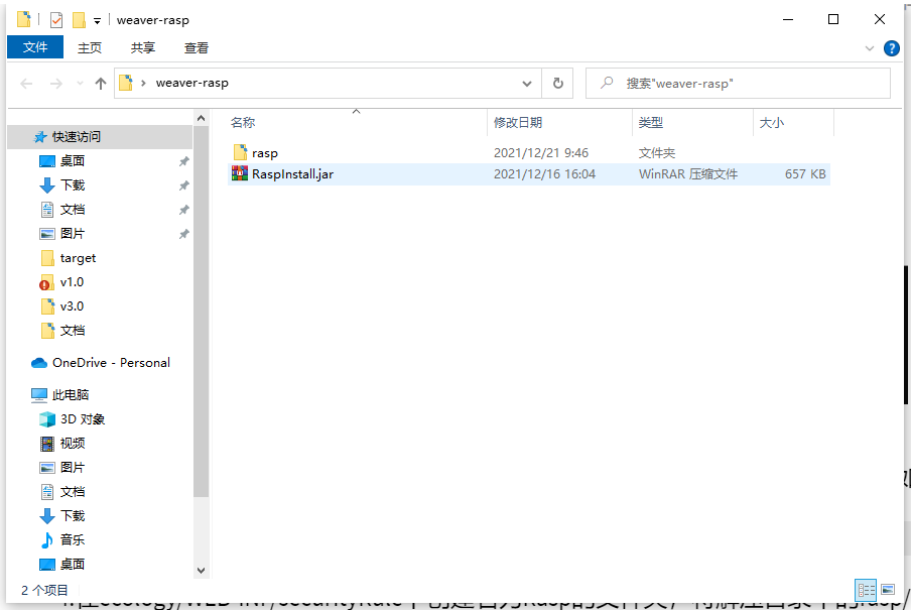
4. 在 ecology/WEB-INF/securityRule 下创建名为 Rasp 的文件夹，将解压目录中的 rasp/plugins/official.js 文件复制到该文件夹下

1. cd /home/weaver/ecology/WEB-INF/securityRule/
2. mkdir Rasp
3. cp /opt/rasp/weaver-rasp/rasp/plugins/official.js /home/weaver/ecology/WEB-INF/securityRule/Rasp

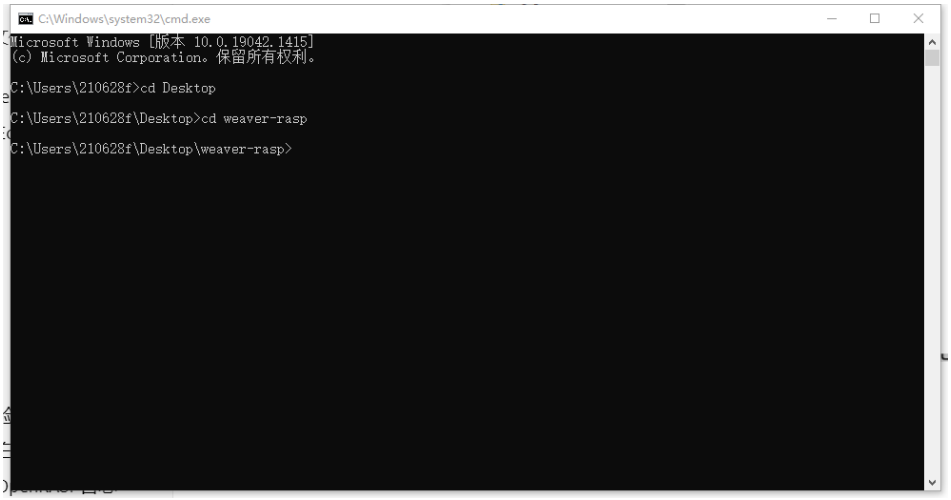
5. 安装步骤完毕，重启 OA 服务后 OpenRASP 生效

Windows

1. 将 weaver-rasp.zip 解压至服务器任意目录

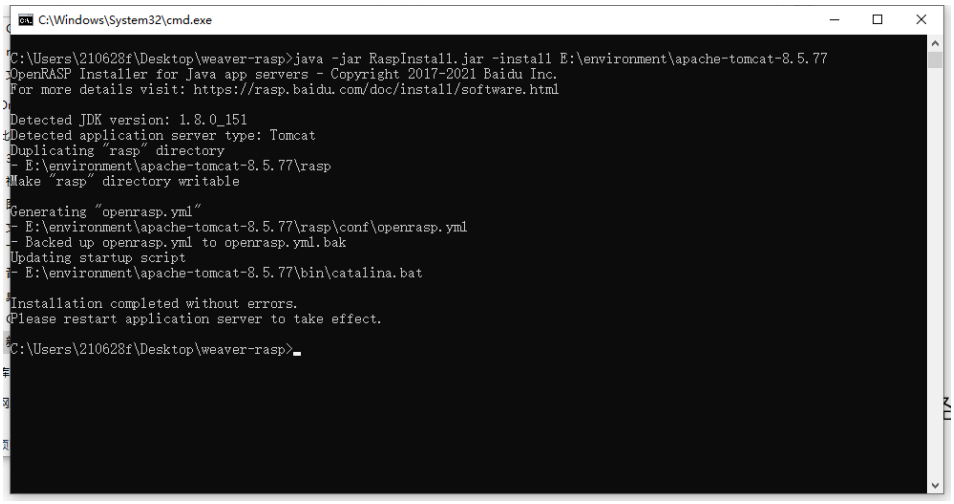


2. 通过 cmd，进入解压后的目录

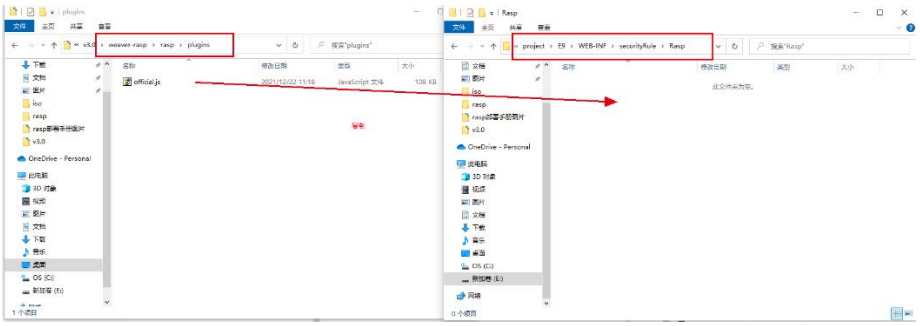


3. 在 CMD 中执行安装命令，将命令中的<tomcat_root>替换为真实的 tomcat 绝对路径

```
1. D:\weaver\Resin\ java -jar RaspInstall.jar -install <tomcat_root>
```



4. 在 ecology/WEB-INF/securityRule 下创建名为 Rasp 的文件夹，将解压目录中的 rasp/plugins/official.js 文件复制到该文件夹下



5. 安装步骤完毕，重启 OA 服务后 OpenRASP 生效

Weblogic 12c（安装包安装）

Linux

1. 将 weaver-rasp.zip 解压至服务器任意目录

1. unzip weaver-rasp.zip

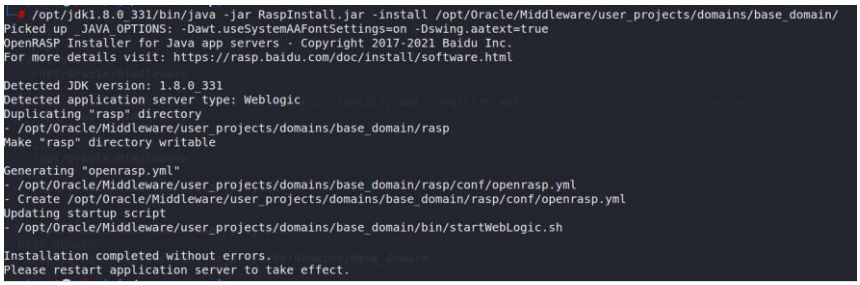
```
└─# unzip weaver-rasp.zip
Archive:  weaver-rasp.zip
   creating: weaver-rasp/
   creating: weaver-rasp/rasp/
   creating: weaver-rasp/rasp/conf/
  inflating: weaver-rasp/rasp/conf/openrasp.yml
   creating: weaver-rasp/rasp/plugins/
  inflating: weaver-rasp/rasp/plugins/official.js
  inflating: weaver-rasp/rasp/rasp-engine.jar
  inflating: weaver-rasp/rasp/rasp.jar
  inflating: weaver-rasp/RaspInstall.jar
```

2. 进入解压后的目录，将<解压位置>替换为真实路径

1. cd <解压位置>/weaver-rasp/

3. 执行安装命令，将命令中的<domain_root>替换为真实的 weblogic 里 domain 的绝对路径，如：/opt/Oracle/Middleware/user_projects/domains/base_domain/

1. java -jar RaspInstall.jar -install < domain_root >



4. 在 ecology/WEB-INF/securityRule 下创建名为 Rasp 的文件夹，将解压目录中的 rasp/plugins/official.js 文件复制到该文件夹下

1. cd /home/weaver/ecology/WEB-INF/securityRule/

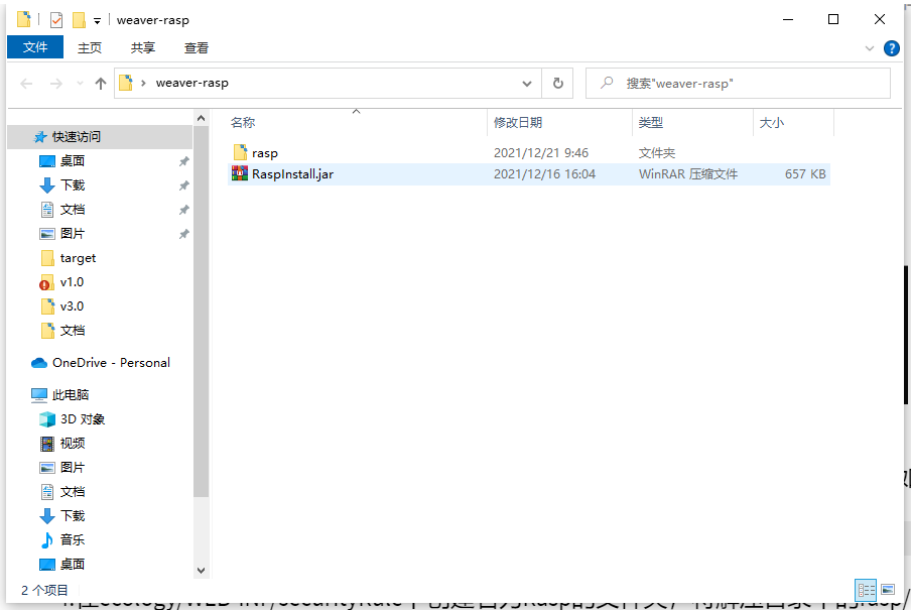
2. mkdir Rasp

3. cp /opt/rasp/weaver-rasp/rasp/plugins/official.js /home/weaver/ecology/WEB-INF/securityRule/Rasp

5. 安装步骤完毕，重启 OA 服务后 OpenRASP 生效

Windows

1. 将 weaver-rasp.zip 解压至服务器任意目录

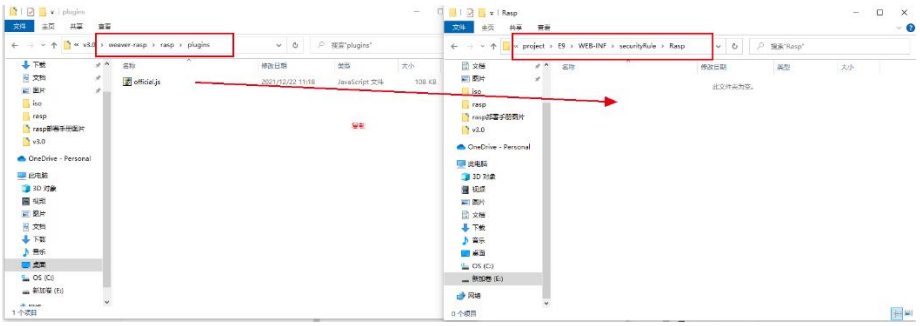


2. 通过 cmd，进入解压后的目录

3. 在 CMD 中执行安装命令，将命令中的<domain_root>替换为真实的 weblogic 里 domain 的绝对路径

1. java -jar RaspInstall.jar -install <domain_root>

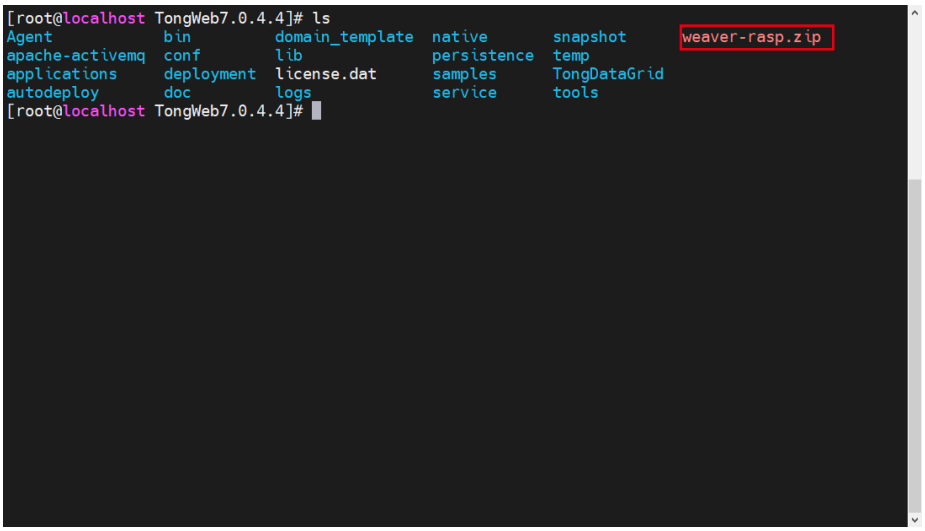
4. 在 ecology/WEB-INF/securityRule 下创建名为 Rasp 的文件夹，将解压目录中的 rasp/plugins/official.js 文件复制到该文件夹下



5. 安装步骤完毕，重启 OA 服务后 OpenRASP 生效

东方通 7（手动安装）

1. 将 weaver-rasp.zip 放在东方通根目录下



2. 解压 weaver-rasp.zip，生成 weaver-rasp 文件

```
1. unzip weaver-rasp.zip
```

```
[root@localhost TongWeb7.0.4.4]# ls
Agent           bin         domain_template  native       snapshot      weaver-rasp.zip
apache-activemq  conf        lib              persistence  temp
applications    deployment  license.dat      samples      TongDataGrid
autodeploy      doc         logs             service      tools
[root@localhost TongWeb7.0.4.4]# unzip weaver-rasp.zip
Archive:  weaver-rasp.zip
   creating: weaver-rasp/
   creating: weaver-rasp/rasp/
   creating: weaver-rasp/rasp/conf/
  inflating: weaver-rasp/rasp/conf/openrasp.yml
   creating: weaver-rasp/rasp/plugins/
  inflating: weaver-rasp/rasp/plugins/official.js
  inflating: weaver-rasp/rasp/rasp-engine.jar
  inflating: weaver-rasp/rasp/rasp.jar
  inflating: weaver-rasp/RaspInstall.jar
[root@localhost TongWeb7.0.4.4]# ls
Agent           bin         domain_template  native       snapshot      weaver-rasp
apache-activemq  conf        lib              persistence  temp          weaver-rasp.zip
applications    deployment  license.dat      samples      TongDataGrid
autodeploy      doc         logs             service      tools
[root@localhost TongWeb7.0.4.4]#
```

3. 将 rasp 文件夹从 weaver-rasp 文件夹内移出

1. mv weaver-rasp/rasp ./

```
Agent           bin         domain_template  native       snapshot      weaver-rasp.zip
apache-activemq  conf        lib              persistence  temp
applications    deployment  license.dat      samples      TongDataGrid
autodeploy      doc         logs             service      tools
[root@localhost TongWeb7.0.4.4]# unzip weaver-rasp.zip
Archive:  weaver-rasp.zip
   creating: weaver-rasp/
   creating: weaver-rasp/rasp/
   creating: weaver-rasp/rasp/conf/
  inflating: weaver-rasp/rasp/conf/openrasp.yml
   creating: weaver-rasp/rasp/plugins/
  inflating: weaver-rasp/rasp/plugins/official.js
  inflating: weaver-rasp/rasp/rasp-engine.jar
  inflating: weaver-rasp/rasp/rasp.jar
  inflating: weaver-rasp/RaspInstall.jar
[root@localhost TongWeb7.0.4.4]# ls
Agent           bin         domain_template  native       snapshot      weaver-rasp
apache-activemq  conf        lib              persistence  temp          weaver-rasp.zip
applications    deployment  license.dat      samples      TongDataGrid
autodeploy      doc         logs             service      tools
[root@localhost TongWeb7.0.4.4]# mv weaver-rasp/rasp ./
[root@localhost TongWeb7.0.4.4]# ls
Agent           bin         domain_template  native       service       tools
apache-activemq  conf        lib              persistence  snapshot      weaver-rasp
applications    deployment  license.dat      rasp         temp          weaver-rasp.zip
autodeploy      doc         logs             samples      TongDataGrid
[root@localhost TongWeb7.0.4.4]#
```

4. 进入 bin 文件夹内，修改 external.vmoptions

1. cd bin
2. vim external.vmoptions

```
#jvm_options
-Xmx2048m
-Xms1024m
-XX:CICompilerCount=6
-server
-XX:+UnlockDiagnosticVMOptions
-XX:+LogVMOutput
-Djava.io.tmpdir=${TongWeb_Base}/temp
-Duser.dir=${TongWeb_Base}/bin
-Djava.security.egd=file:/dev/./urandom
-XX:LogFile=${TongWeb_Base}/logs/jvm.log
-XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=${TongWeb_Base}/logs/heap${sysdate}.hprof
-Djava.security.policy=${TongWeb_Base}/conf/tongweb.policy
-Djava.rmi.server.RMIClassLoaderSpi=com.tongweb.server.TongWebRMIClassLoader
-Djava.util.logging.manager=com.tongweb.log.TongwebLogManager
-javaagent:${TongWeb_Home}/lib/ejb-agent.jar
-Djava.awt.headless=true
-Dibm.stream.nio=true
-Djava.net.preferIPv4Stack=false
-XX:MaxMetaspaceSize=1024m
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/jdk.internal.ref=ALL-UNNAMED
--add-modules=ALL-MODULE-PATH
--module-path=${JAVA_ENDORSED_DIRS}
--upgrade-module-path=${JAVA_ENDORSED_DIRS}/
"external.vmoptions" 73L, 2285C 已写入                                  18,1          顶端
```

5. 在-javaagent:${.... 一行下增加如下语句后保存

1. -javaagent:${TongWeb_Home}/rasp/rasp.jar

```
#jvm_options
-Xmx2048m
-Xms1024m
-XX:CICompilerCount=6
-server
-XX:+UnlockDiagnosticVMOptions
-XX:+LogVMOutput
-Djava.io.tmpdir=${TongWeb_Base}/temp
-Duser.dir=${TongWeb_Base}/bin
-Djava.security.egd=file:/dev/./urandom
-XX:LogFile=${TongWeb_Base}/logs/jvm.log
-XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=${TongWeb_Base}/logs/heap${sysdate}.hprof
-Djava.security.policy=${TongWeb_Base}/conf/tongweb.policy
-Djava.rmi.server.RMIClassLoaderSpi=com.tongweb.server.TongWebRMIClassLoader
-Djava.util.logging.manager=com.tongweb.log.TongwebLogManager
-javaagent:${TongWeb_Home}/lib/ejb-agent.jar
-javaagent:${TongWeb_Home}/rasp/rasp.jar
-Djava.awt.headless=true
-Dibm.stream.nio=true
-Djava.net.preferIPv4Stack=false
-XX:MaxMetaspaceSize=1024m
--add-opens=java.base/java.lang=ALL-UNNAMED
--add-opens=java.base/jdk.internal.ref=ALL-UNNAMED
--add-modules=ALL-MODULE-PATH
--module-path=${JAVA_ENDORSED_DIRS}
"external.vmoptions" 74L, 2326C 已写入                                  18,40         顶端
```

6. 安装步骤完毕，重启 OA 服务后 OpenRASP 生效

宝兰德（手动安装）

1. 将 weaver-rasp.zip 放在宝兰德根目录下

```
[root@localhost BES-9.5.2.4160]# ls
bin   deployments  hotdeploy  license  modules  repository  weaver-rasp.zip
conf  docs         lib        logs     patch    samples     work
[root@localhost BES-9.5.2.4160]#
```

2. 解压 weaver-rasp.zip，生成 weaver-rasp 文件

1. unzip weaver-rasp.zip

```
[root@localhost BES-9.5.2.4160]# ls
bin   deployments  hotdeploy  license  modules  repository  weaver-rasp      work
conf  docs         lib        logs     patch    samples     weaver-rasp.zip
[root@localhost BES-9.5.2.4160]#
```

3. 将 rasp 文件夹从 weaver-rasp 文件夹内移出

1. mv weaver-rasp/rasp ./

```
conf  docs         lib        logs     patch    samples     weaver-rasp.zip
[root@localhost BES-9.5.2.4160]# mv weaver-rasp/rasp/ ./
[root@localhost BES-9.5.2.4160]# ls
bin   deployments  hotdeploy  license  modules  rasp        samples      weaver-rasp.zip
conf  docs         lib        logs     patch    repository  weaver-rasp  work
[root@localhost BES-9.5.2.4160]# clear
[root@localhost BES-9.5.2.4160]# ls
bin   deployments  hotdeploy  license  modules  rasp        samples      weaver-rasp.zip
conf  docs         lib        logs     patch    repository  weaver-rasp  work
[root@localhost BES-9.5.2.4160]#
```

4. 进入 conf 文件夹，编辑 server.conf 文件

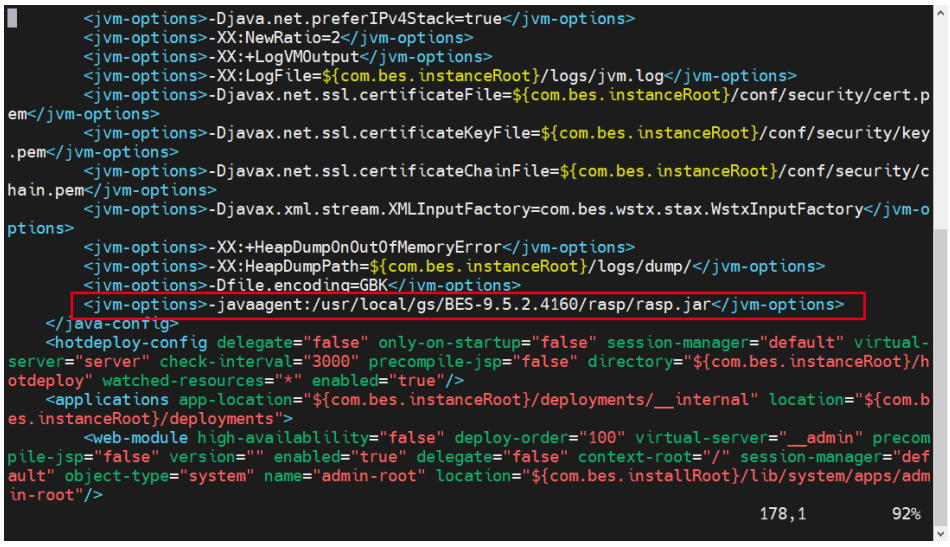
1. cd conf

2. vim server.conf

```
<?xml version='1.0' encoding='UTF-8'?>
<server>
    <ejb-container http-channel-enabled="true" enabled="true">
        <mdb-container instance-limit="10" resource-adapter="jmsra" fail-on-unknow-activation-spec="true" activation-spec-class="com.bes.mq.ra.BESMQActivationSpec" enabled="true" message-listener-interface="javax.jms.MessageListener"/>
        <singleton-container access-timeout-in-seconds="30"/>
        <stateful-container frequency-in-seconds="60" session-store="${com.bes.instanceRoot}/repository/session" timeout-in-seconds="1200" access-timeout-in-seconds="30" bulk-passivate="100" capacity="1000"/>
        <ejb-listener address="0.0.0.0" min-spare-threads="8" tcp-no-delay="true" work-threads="200" enabled="true" mode="BIO" receive-buffer-size="-1" backlog="200" thread-rate="2" max-idle-time="120" port="3000" request-timeout="60" name="ejb-listener-1" send-buffer-size="-1" max-threads="128" max-queue-size="4096" request-timeout-warning="0" request-handle-buffer-size="8192"/>
        <stateless-container max-age-in-seconds="0" min-size="0" max-size="10" idle-timeout-in-seconds="0" access-timeout-in-seconds="30"/>
    </ejb-container>
    <web-container enabled="true">
        <session-managers>
            <session-manager name="default" class-name="com.bes.enterprise.webtier.session.DefaultManager">
                <manager-properties>
                    <property name="maxInactiveInterval" value="1800"/>
                    <property name="reapInterval" value="60"/>
                    <property name="maxActiveSessions" value="-1"/>
                </manager-properties>
"server.config" 207L, 27037C                                          1,1           顶端
```

5. 在<jvm-options>-Dfile.encoding=GBK</jvm-options>下增加如下内容

1. <jvm-options>-javaagent:/usr/local/gs/BES-9.5.2.4160/rasp/rasp.jar</jvm-options>



```
        <jvm-options>-Djava.net.preferIPv4Stack=true</jvm-options>
        <jvm-options>-XX:NewRatio=2</jvm-options>
        <jvm-options>-XX:+LogVMOutput</jvm-options>
        <jvm-options>-XX:LogFile=${com.bes.instanceRoot}/logs/jvm.log</jvm-options>
        <jvm-options>-Djavax.net.ssl.certificateFile=${com.bes.instanceRoot}/conf/security/cert.pem</jvm-options>
        <jvm-options>-Djavax.net.ssl.certificateKeyFile=${com.bes.instanceRoot}/conf/security/key.pem</jvm-options>
        <jvm-options>-Djavax.net.ssl.certificateChainFile=${com.bes.instanceRoot}/conf/security/chain.pem</jvm-options>
        <jvm-options>-Djavax.xml.stream.XMLInputFactory=com.bes.wstx.stax.WstxInputFactory</jvm-options>
        <jvm-options>-XX:+HeapDumpOnOutOfMemoryError</jvm-options>
        <jvm-options>-XX:HeapDumpPath=${com.bes.instanceRoot}/logs/dump/</jvm-options>
        <jvm-options>-Dfile.encoding=GBK</jvm-options>
        <jvm-options>-javaagent:/usr/local/gs/BES-9.5.2.4160/rasp/rasp.jar</jvm-options>
    </java-config>
    <hotdeploy-config delegate="false" only-on-startup="false" session-manager="default" virtual-server="server" check-interval="3000" precompile-jsp="false" directory="${com.bes.instanceRoot}/hotdeploy" watched-resources="*" enabled="true"/>
    <applications app-location="${com.bes.instanceRoot}/deployments/__internal" location="${com.bes.instanceRoot}/deployments">
        <web-module high-availablility="false" deploy-order="100" virtual-server="__admin" precompile-jsp="false" version="" enabled="true" delegate="false" context-root="/" session-manager="default" object-type="system" name="admin-root" location="${com.bes.installRoot}/lib/system/apps/admin-root"/>
                                                                      178,1         92%
```

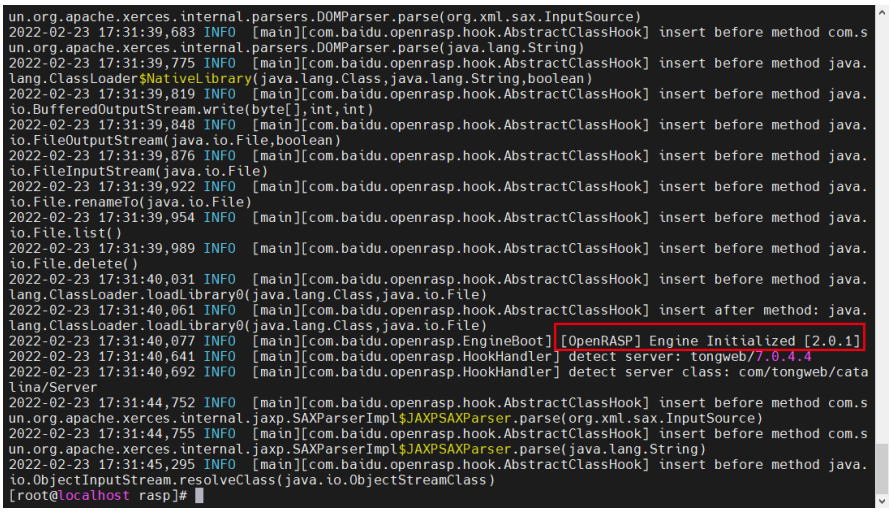
6. 安装步骤完毕，重启 OA 服务后 OpenRASP 生效

验证部署结果

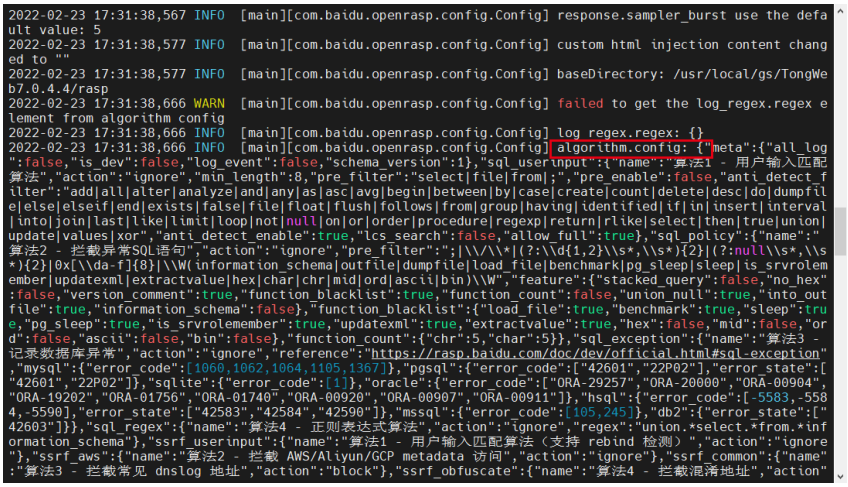
方式一：检查日志验证部署

重启服务后，在 rasp 运行日志中查找：

1. [OpenRASP] Engine Initialized 字样，如果找到，则说明 OpenRasp 成功运行



2. algorithm.config:字样，如果找到，则说明 OpenRasp 防护插件成功加载



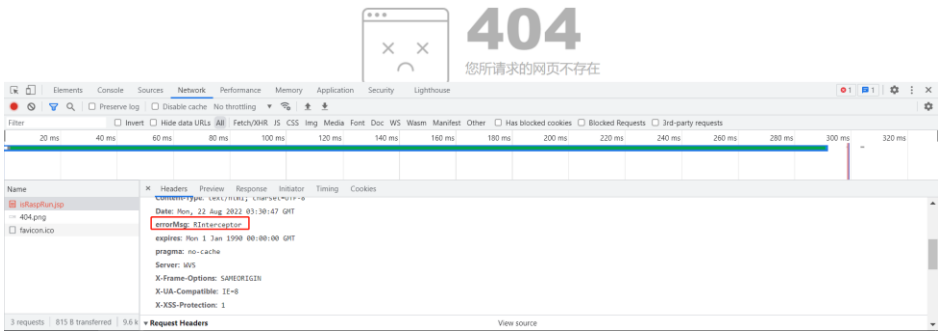
如上两条字样能同时找到，则 rasp 部署成功

rasp 运行日志地址：中间件根目录/rasp/logs/rasp/rasp.log

比如：/oa/weaver/Resin/rasp/logs/rasp/rasp.log

方式二：通过访问页面校验部署

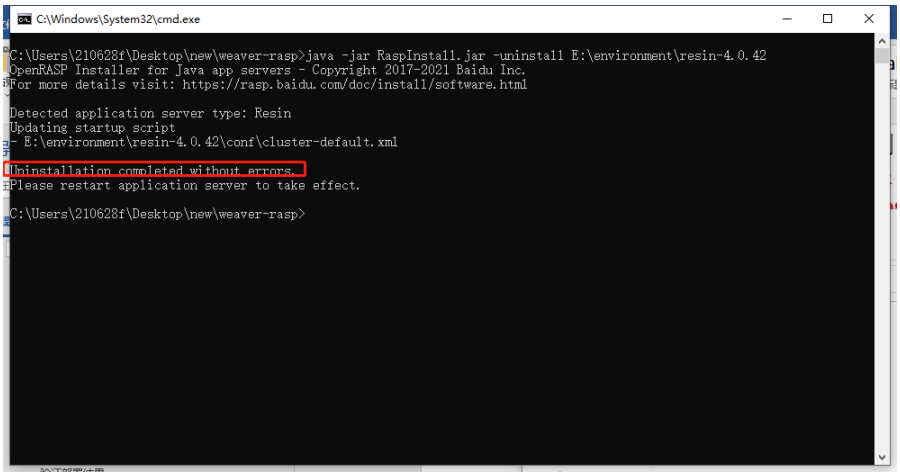
1. 安装完 RASP 后使用 sysadmin 登录 OA，然后访问/security/monitor/isRaspRun.jsp，如果显示是 404 且控制台头部信息中有 errorMsg:RInterceptor 字样则表示安装成功。



卸载 OpenRASP

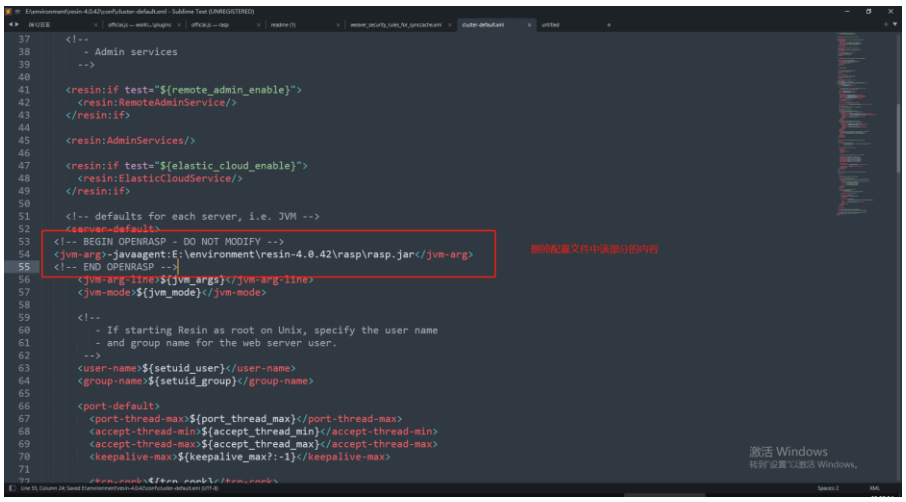
使用安装包安装 RASP 后卸载

1. 进入 RASP 安装包，执行命令：java -jar RaspInstall.jar -uninstall <Resin 路径>后重启 OA 即可



手动安装后卸载

1. 编辑带有 rasp 参数的配置文件，删除增加的 rasp 相关参数后重启 OA 即可

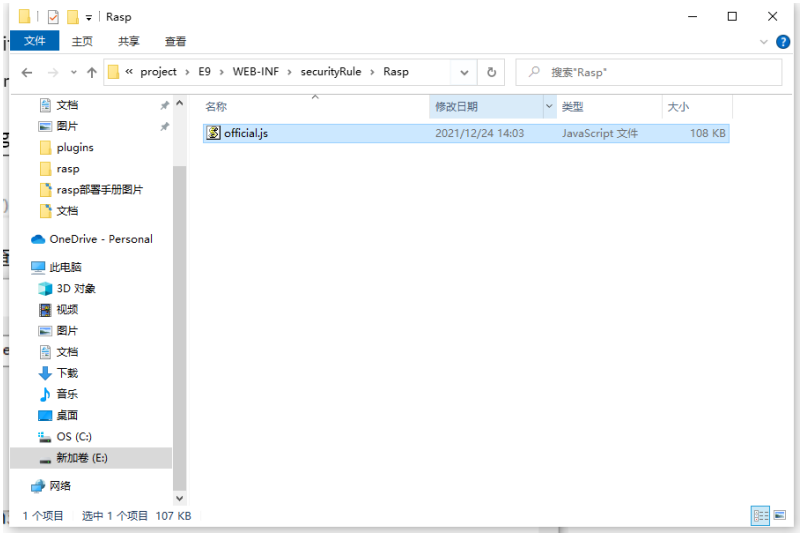


白名单配置

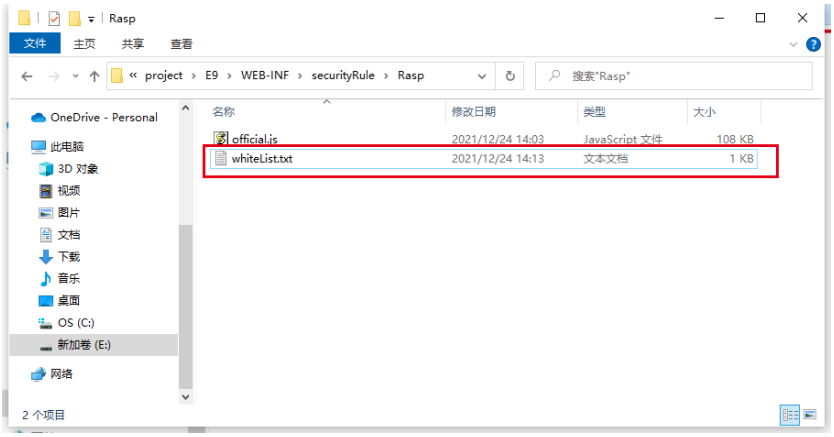
文件上传路径白名单配置

OpenRASP 默认拦截后缀为 jsp、php 的上传文件。如果客户系统确实需要上传此类文件，那么可通过将上传文件的存储路径加入白名单来避免请求被拦截。方法如下：

1. 打开 ecology/WEB-INF/securityRule/Rasp 文件夹：

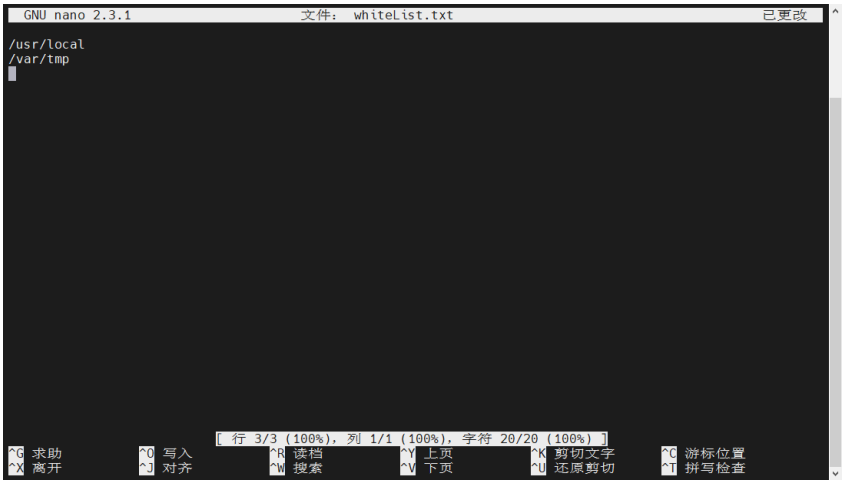


2. 创建 whiteList.txt 文件（**注意大小写**）



3. 将系统中上传文件的目录路径按照格式添加到 whiteList.txt 文件中。具体路径格式如下：
   a) 多个路径使用**换行符**分隔
   b) 路径末尾除换行符之外不能添加任何符号
   c) Windows 系统中**磁盘号要大写，分隔符使用/号，有且只有一个**，如下图：

   d) Linux 系统中**分隔符使用/号，有且只有一个** ，如下图：

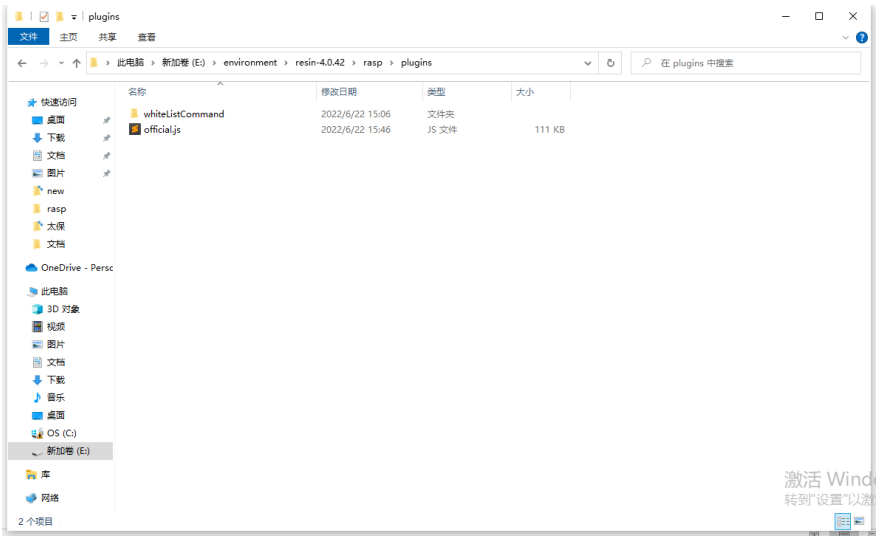


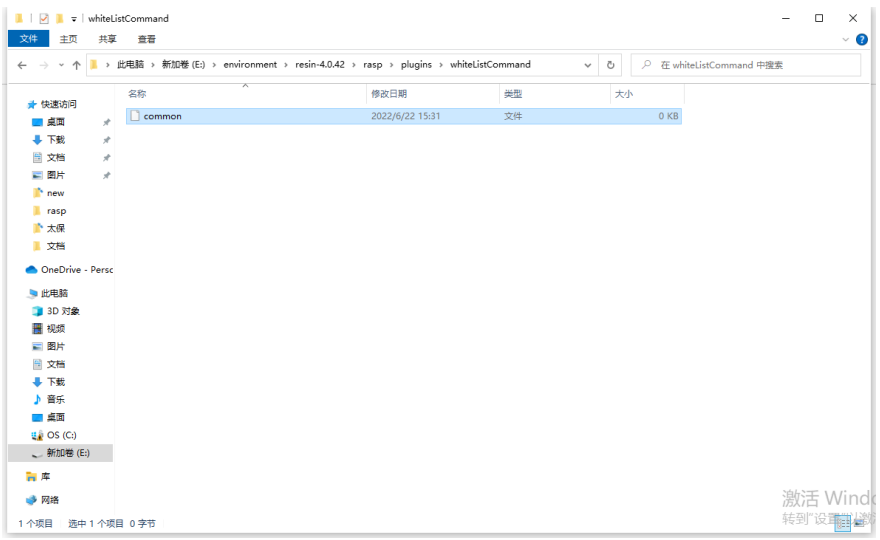
命令执行白名单配置

- 由于系统中会存在二次开发的功能，这些功能可能会包含执行系统命令，从而产生被 RASP 拦截后功能异常的问题，因此开放命令白名单配置，用户可以自己添加命令执行的白名单。由于该配置技术性较强，因此建议技术人员来配置。

方法 1：通用性配置

- 通用性配置是指配置完后所有的 java 类都可以执行该系统命令。方法如下：
  1. 进入 ecology/WEB-INF/securityRule/Rasp，新建文件夹 whiteListCommand（注意大小写），进入 whiteListCommand 文件夹，新建文件 common





  2. 将需要加白的命令以正则表达式的方式写入该文件，多个命令合并成一条正则表达式。完成之后保存退出即可。

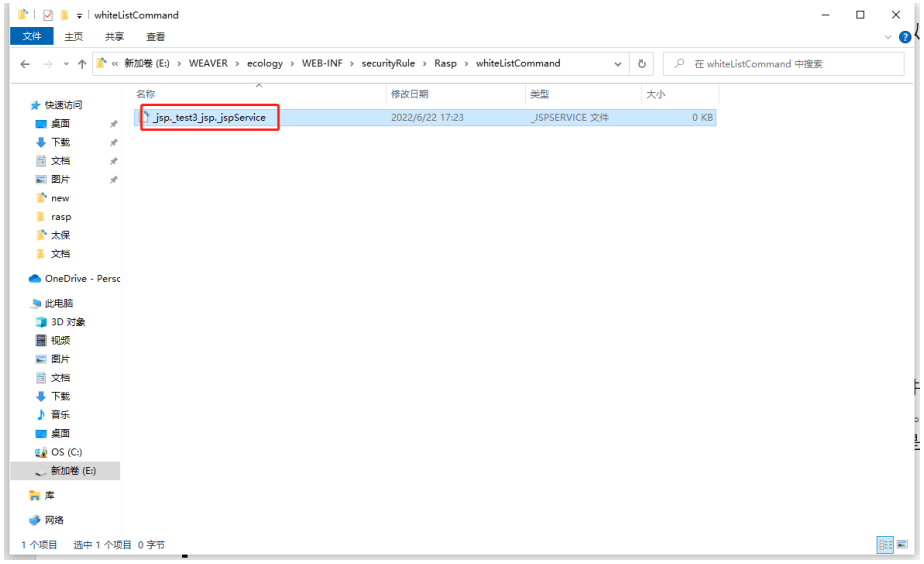
方法 2：方法层面配置

- 方法层面配置是指配置完后只有指定类的指定方法可以执行该系统命令。如果某条命令只在某个方法中被调用，出于安全性考虑，建议使用该方法配置白名单。方法如下：
  1. 进入 ecology/WEB-INF/securityRule/Rasp，打开 official.js 文件，搜索 command_comm，找到如下内容：

  2. 在此处增加日志输出语句：plugin.log(params) 后保存

  3. 触发问题功能，使问题复现，然后查看 resin/rasp/logs/plugin/plugin.log 文件，可以看到 RASP 将拦截的命令和堆栈都打印了出来

4. 找到需要加白的类和方法后，进入 ecology/WEB-INF/securityRule/Rasp 下创建文件夹 whiteListCommand，进入 whiteListCommand 文件夹创建以加白堆栈为名的文件。（比如要加白上图中的 _jsp._test3_jsp._jspService，那么创建的文件名就是 _jsp._test3_jsp._jspService）



5. 将命令对应的正则放入该文件内第一行，如果有多条命令，则合并为一条正则表达式。完成之后保存退出即可。

说明

1. 如果不清楚具体被拦截的是哪条命令，那么可以通过在 official.js 中增加日志输出的方式来确认，详细步骤参考**方法层面配置中的第 3 步**
2. 如果增加了日志输出语句，在配置完白名单后要将该语句删除，避免产生多余的日志信息，从而导致日志文件过大
3. 可以同时使用**通用性配置**和**方法层面配置**

OpenRASP 日志

安装完成后，在中间件的根目录下会生成名称为 rasp 的文件夹。收集日志时，需要将该 rasp 文件夹下的 logs 文件夹打包压缩。各个日志及功能如下：

文件名	文件内容
plugin/plugin-DATE.log	检测插件的日志，e.g 插件异常、插件调试输出
rasp/rasp-DATE.log	rasp agent 调试日志
alarm/alarm-DATE.log	攻击报警日志，JSON 格式，一行一个
policy_alarm/policy_alarm-DATE.log	安全基线检查报警日志，JSON 格式，一行一个

分析攻击行为时可重点关注 alarm 日志；alarm 日志是 json 格式，建议在文本编辑器里下载格式化 json 的插件，然后将对应的 json 格式化后输出，比如：