DSProtector使用说明

产品简介

DSProtector是配合 Virbox Protector 对资源或脚本文件进行加密的工具。

常见文件列表

以下列表为目前测试涉及到的程序或场景，若有未涉及到的情况可联系**深思客服**进行详细的咨询。

Windows平台

类型	描述	是否支持
python	py、pyc	支持
java	jar、war(class)	支持
php	php	支持
go	.go	支持
Lua	.lua	支持
Perl	.pl	支持
R	.R	支持
Ruby	.rb	支持
Unity3D	dll、reS、resources、MP4等资源文件	支持
视频|音频	MP4、MP3	支持
UE4	pak	支持

Linux平台

类型	描述	是否支持
python	py、pyc	支持
java	jar、war(class)	支持
php	php	支持
go	.go	不支持
Lua	.lua	支持
Perl	.pl	支持
R	.R	不支持
Ruby	.rb	支持
Unity3D	dll、reS、resources、MP4等资源文件	支持
视频|音频	MP4、MP3	支持

ARM Linux平台

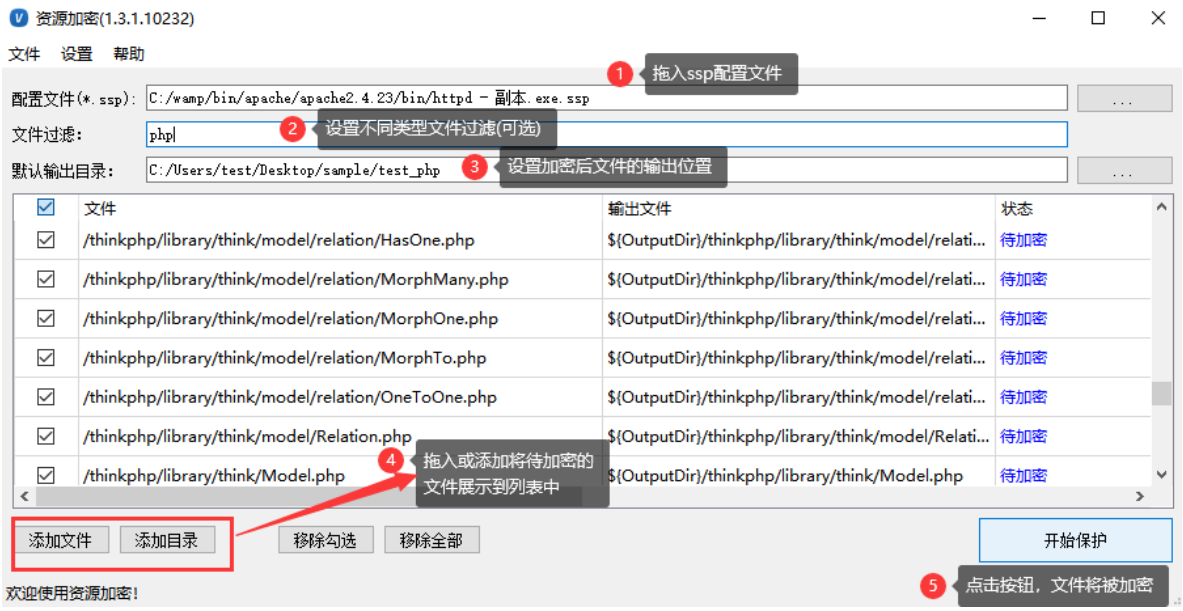
类型	描述	是否支持
python	py、pyc	支持
java	jar、war(class)	支持
php	php	支持
Lua	.lua	支持

使用场景

界面工具使用

通过Virbox Protector工具打开DSProtector，DSProtector主界面的配置文件会自动填写。

1、DSProtector主界面及一些控件的功能如图所示：



2、“默认输出目录”功能使用方式

- 若要将文件设置输出到指定位置，需要先设置“默认输出目录”选项，然后在将文件拖入到DSProtect工具界面的文件列表中，此时“默认输出目录”选项的功能才生效。
  - 举例：

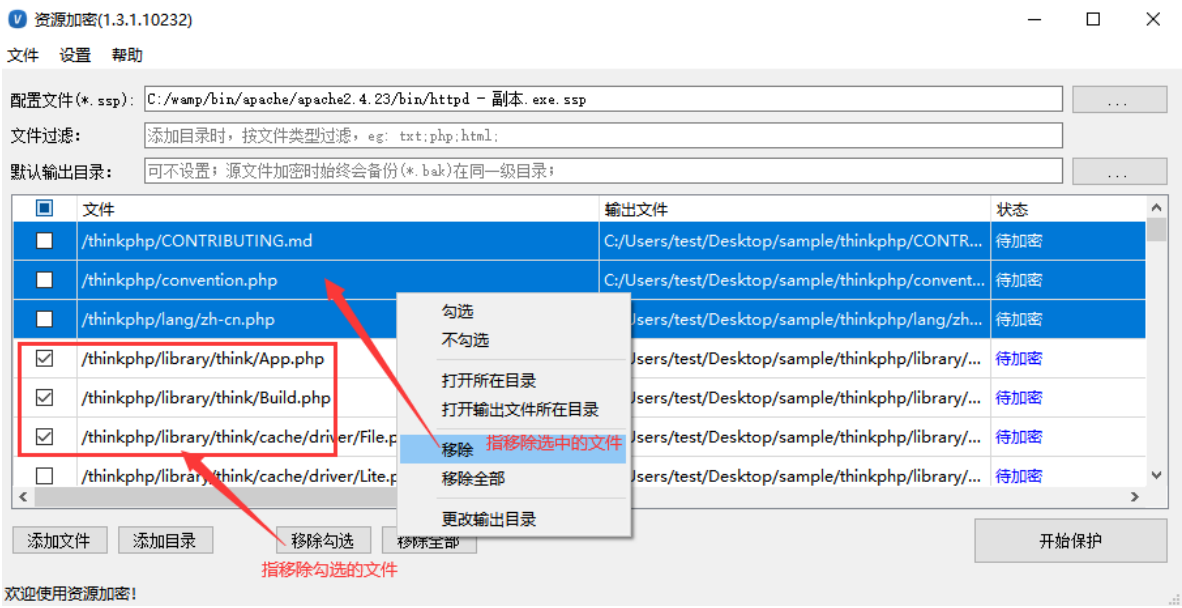
    > 文件目录：/thinkphp/library/think/test.php。
    >
    > 输出文件目录：$(OutputDir)/thinkphp/library/think/test.php。

  - 原始文件不会进行任何更改，加密后的文件在设置的指定目录下。
- 若先拖入文件到DSProtect工具界面的文件列表中，再设置“默认输出目录”，则此时“输出文件”列表下的文件所指路径为当前文件存在的路径。
  - 举例：

    > 文件列表：/thinkphp/library/think/test.php。
    >
    > 输出文件目录：/thinkphp/library/think/test.php。

  - 原始文件备份为test.php.bak，加密后的文件名为：test.php。

3、文件鼠标右键的功能。



命令行工具使用

目前仅Windows和Linux系统支持DS及命令行工具。

命令参数解析

参数	功能
filename	指对单个文件的保护
directory	指对文件目录的保护
-c ssp	指Virbox Protector工具生成的配置文件
-o output	指保护后输出文件路径

使用方法

DSProtector工具主要对PHP、java、Unity3D、python等资源文件进行加密保护。**以Linux平台Unity3D的资源文件为例：**

Unity3D目录结构简介说明

```
PlayOgv-bdwgc-x64_Data
├─Managed
├─MonoBleedingEdge
│  ├─etc
│  │  └─mono
│  └─x86_64
├─Plugins
└─Resources
   ├─level0
  ├─level0.resS
```

- 【必选】使用Virbox Protector工具生成配置文件，并且ds按钮必须打开。

- 打开终端窗口，进入到“dsprotector_con.exe”所在的路径，直接输入“dsprotector_con.exe”运行可查看帮助信息

- 单个资源文件保护，可以先将原始文件改名备份。

> 命令：dsprotector_con.exe ssp.Linux_PlayOgv_x64\PlayOgv-bdwgc-x64_Data\level0.resS -c Linux_PlayOgv_x64.ssp -o ssp.Linux_PlayOgv_x64\PlayOgv-bdwgc-x64_Data\level0_virbox.resS

```
C:\Users\test\Desktop\virboxprotector_standalone_1.4.2.10236_windows_x64\bin>dsprotector_con.exe C:\Users\test\Desktop\sample\ssp.Linux_PlayOgv_x64\PlayOgv-bdwgc-x64_Data\level0.resS.bak -c C:\Users\test\Desktop\sample\Linux_PlayOgv_x64.ssp -o C:\Users\test\Desktop\sample\ssp.Linux_PlayOgv_x64\PlayOgv-bdwgc-x64_Data\level0.reS
dsprotector_con
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


Succeed.
```

- 资源目录保护，可以先将原始资源目录名称改名备份。

> 命令：dsprotector_con.exe ssp.Linux_PlayOgv_x64\PlayOgv-bdwgc-x64_Data\Resources.bak -c Linux_PlayOgv_x64.ssp -o ssp.Linux_PlayOgv_x64\PlayOgv-bdwgc-x64_Data\Resources

```
C:\Users\test\Desktop\virboxprotector_standalone_1.4.2.10236_windows_x64\bin>dsprotector_con.exe C:\Users\test\Desktop\sample\ssp.Linux_PlayOgv_x64\PlayOgv-bdwgc-x64_Data\Resources.bak -c C:\Users\test\Desktop\sample\Linux_PlayOgv_x64.ssp -o C:\Users\test\Desktop\sample\ssp.Linux_PlayOgv_x64\PlayOgv-bdwgc-x64_Data\Resources
dsprotector_con
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


Succeed.
```

命令行错误码列表

错误码	详细信息
Error（0x00000000）	success，成功。
Error（0x00000001）	无效的参数。
Error（0x00000002）	内存不足。
Error（0x00000003）	配置文件错误。
Error（0x00000011）	文件已经加过密。
Error（0x00000012）	读取文件失败，filename文件或directory目录找不到
Error（0x00000013）	写入文件失败，指文件被占用或无权限写入。
Error（0x00000021）	未发现ssp文件。
Error（0x00000022）	配置文件错误。
Error（0x00000023）	ssp文件缺失ds插件配置，或手动将ssp文件中的ds节点给删除了。
Error（0x00000024）	ssp文件ds插件未激活，是指DS按钮未打开。
Error（0x00000025）	ssp文件中缺少ds插件密码节点，没有输入密码或者配置文件中密码节点给删除了。
Error（0x00000031）	文件太小（<=4字节）。

语言类型

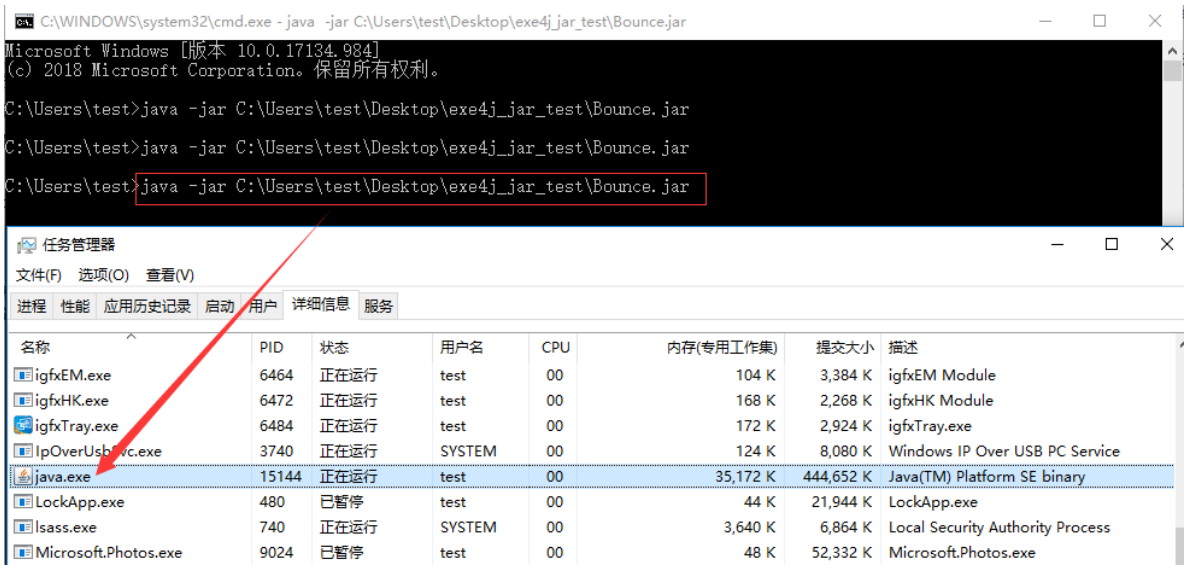
jar文件保护

jar文件的运行方式很多，使用DSProtector工具对文件进行加密，主要是将jar文件和调用jar文件的主程序进行关联。以下介绍几种使用较多的场景操作流程。

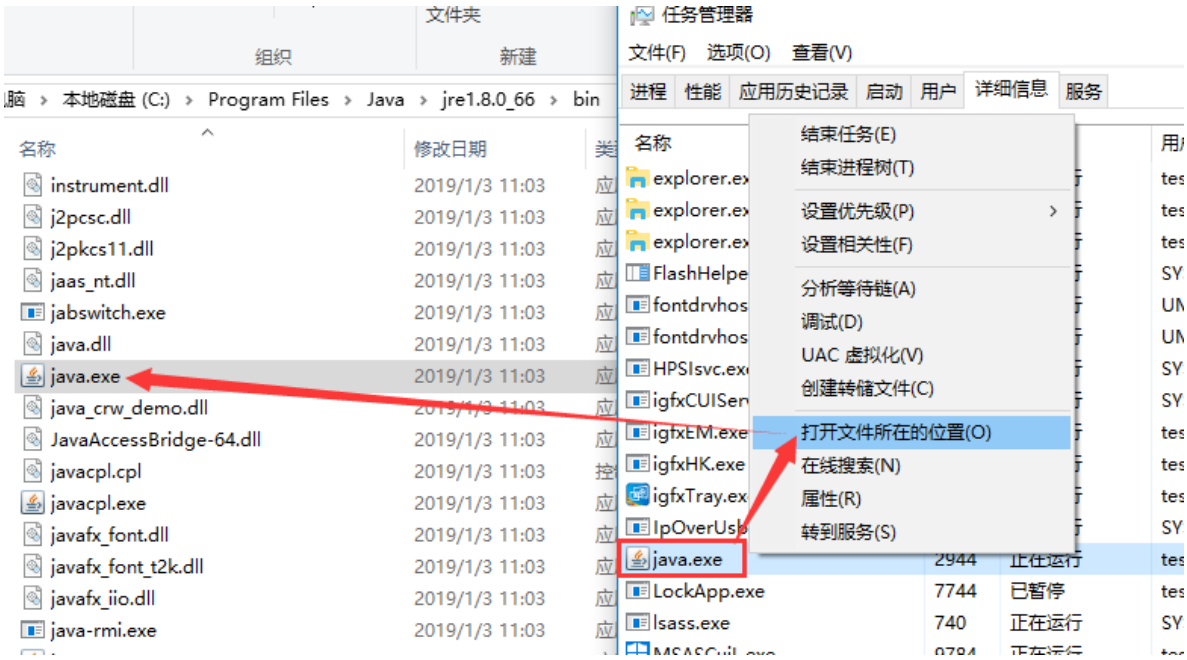
java解释器-Windows

1、使用java解释器直接运行的jar文件。

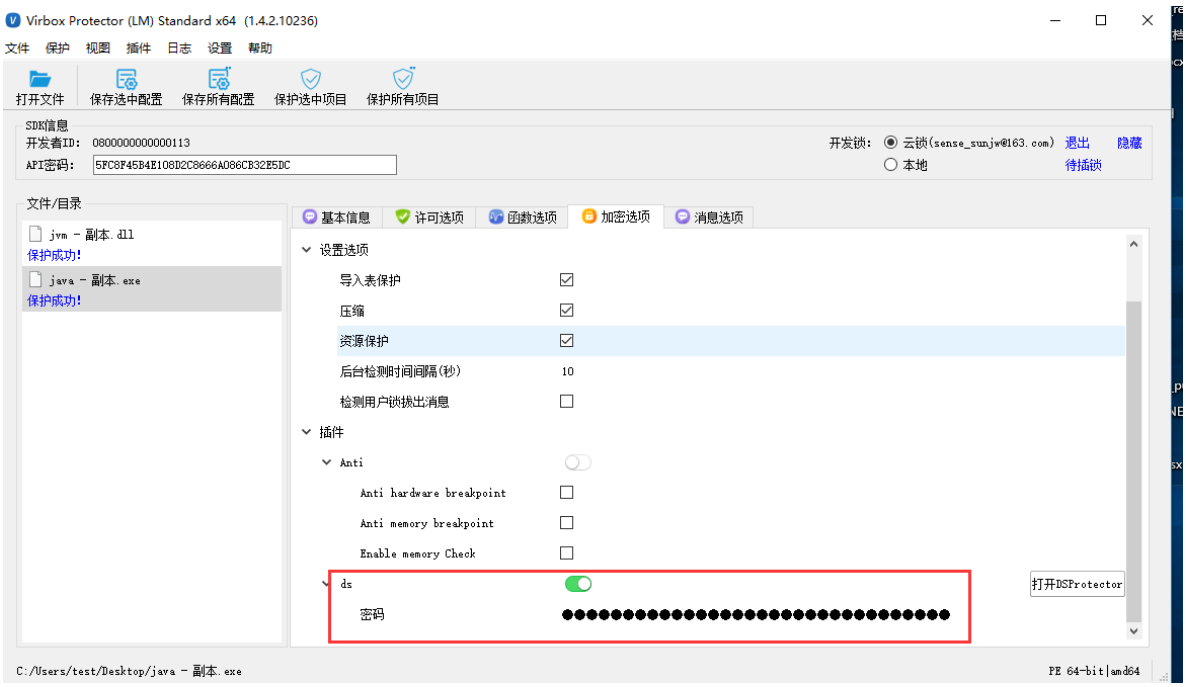
> 命令：java -jar .\exe4j_jar_test\Bounce.jar



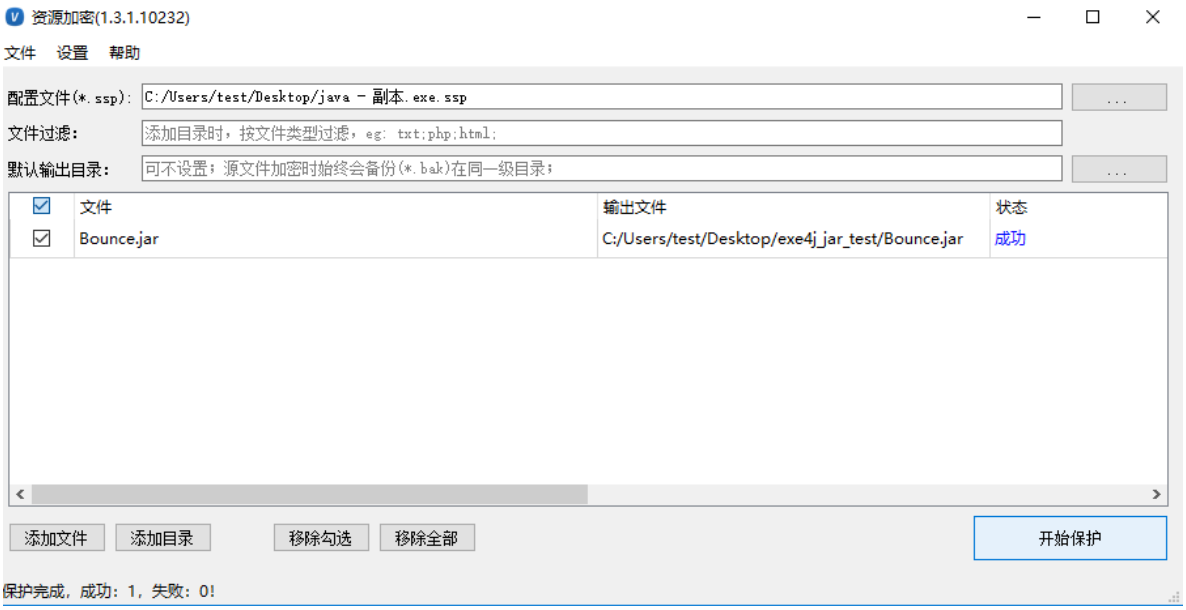
2、通过任务管理器可以看到进程的位置。



3、使用Virbox Protector工具对java.exe进行保护



4、使用DSProtector工具对jar包进行加密



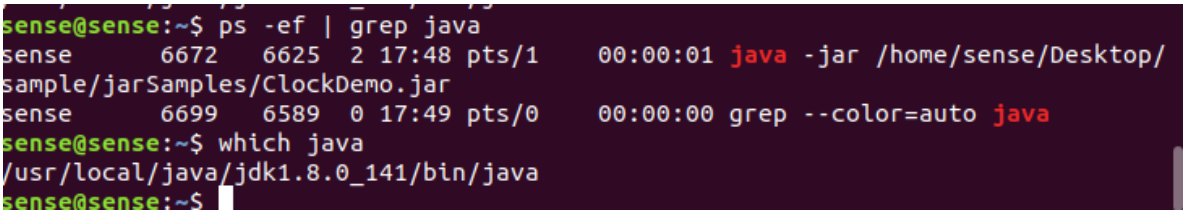
5、将原始文件java.exe备份，将保护后的java.exe替换到原始文件存在的位置，执行运行命令即可正常运行。

java解释器-Linux

1、使用java解释器直接运行的jar文件。

> 命令：java -jar .\exe4j_jar_test\Bounce.jar

2、找到调用jar包文件的主进程java的位置。



3、使用Virbox Protector工具对其加壳保护，选择要保护jar包文件，使用DSProtector工具进行加密即可。

第三方工具exe4j打包jar文件

场景：在Windows系统上，客户使用exe4j将jar包直接转为exe文件，这种情况如何保护才能保证程序安全?

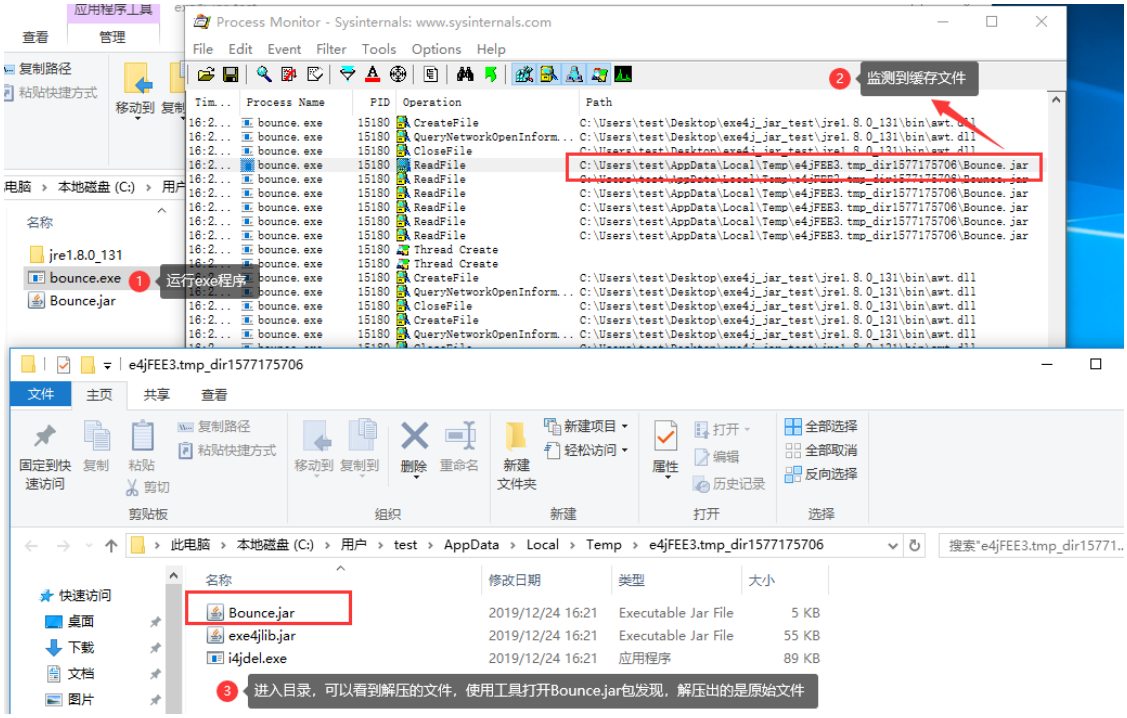
以下两种方案请根据自己实际情况选择。

【注意】任何通过第三方工具打包的exe运行时均有可能产生临时文件，请自己核实具体情况。

方案一

直接使用Virbox Protect工具对exe文件进行加壳保护。

- 优点：方便，快捷。
- 缺点：由于exe运行会产生临时文件，可以通过监测exe找到缓存文件的位置，获取未加密文件。
- 如图所示：



Virbox Protect工具只能对exe进行保护，无法改变exe内部操作，所以无法防止exe运行时产生临时文件。

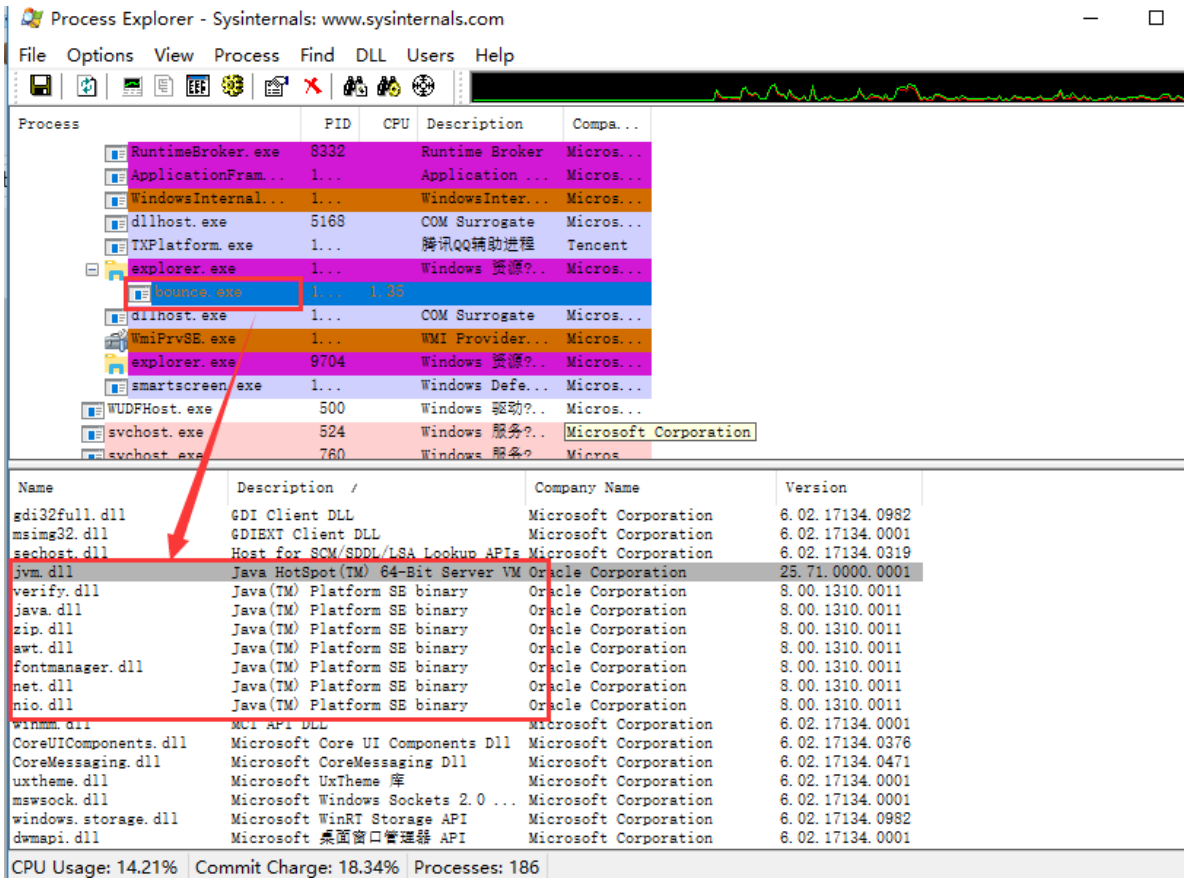
方案二

使用Virbox Protector对所依赖环境的主进程进行加壳保护，再使用DSProtector对jar包进行保护。

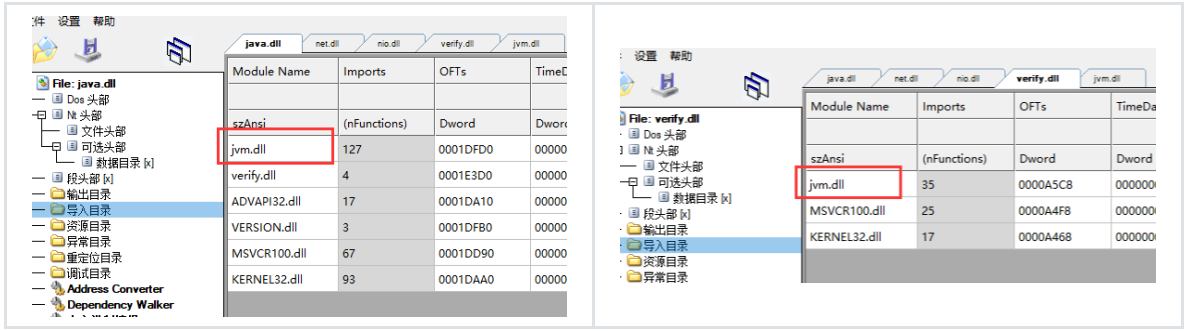
- 优点：exe运行产生的临时文件是使用DSProtector加密保护过的，无法被反编译。
- 缺点：操作流程较麻烦。

1、首先使用启动exe程序，打开任务管理器，可以看到无java.exe进程启动。

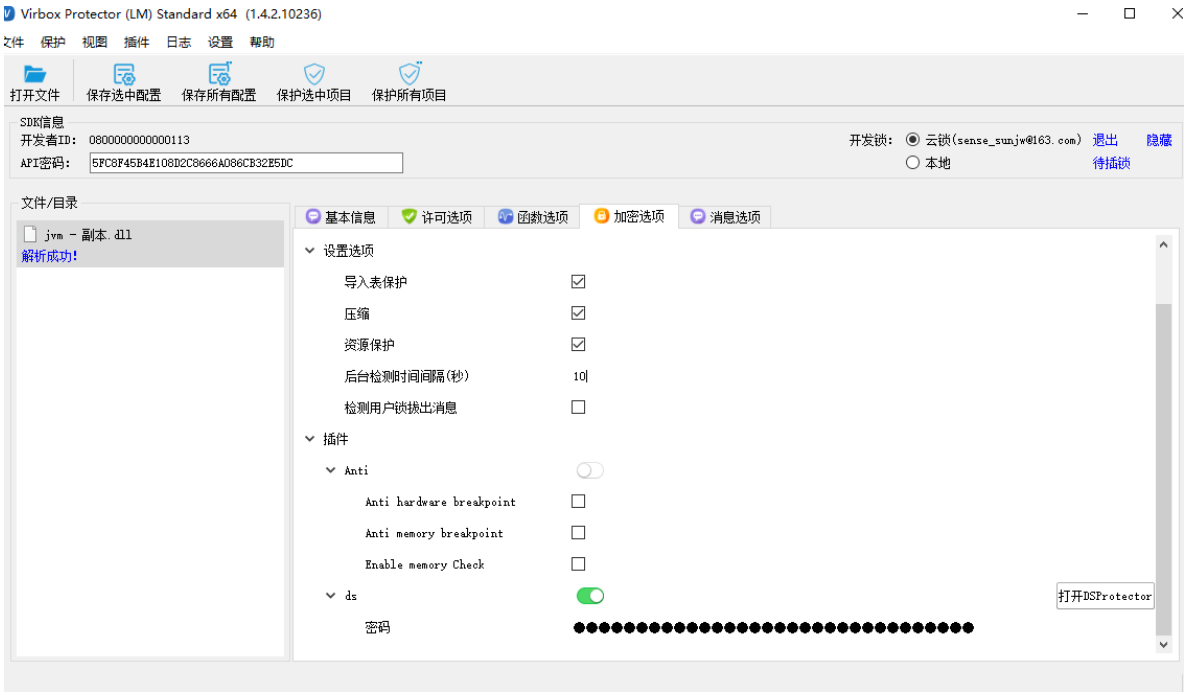
2、使用Procmon.exe和procexp.exe工具对原始exe程序监测，查找其调用的主程序。



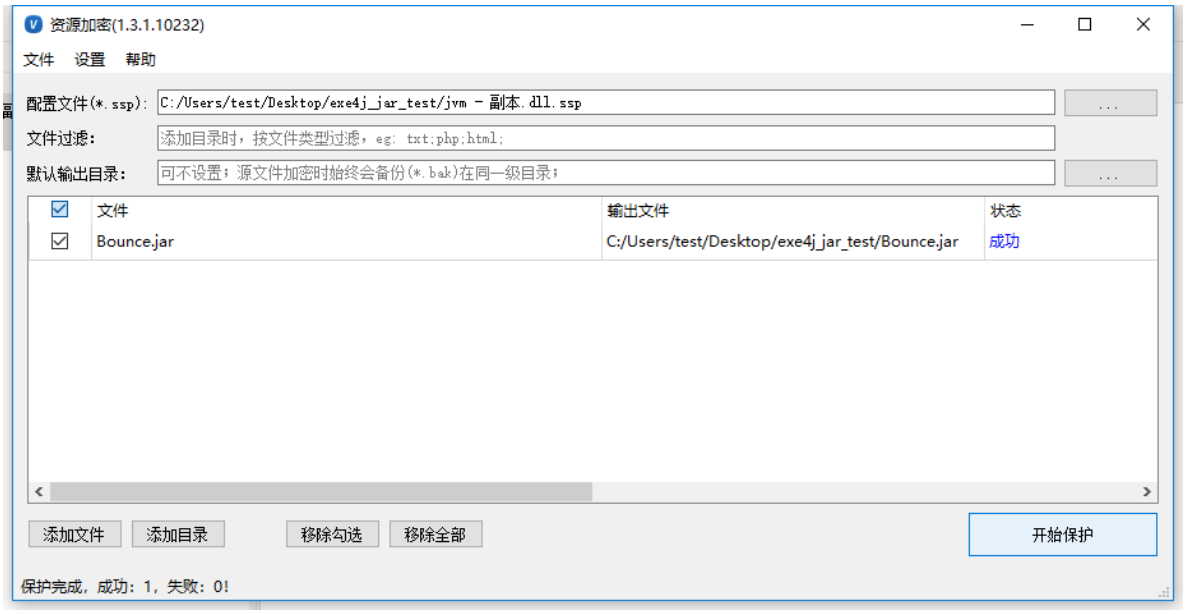
3、将监测到所调用的dll拖入CFF Explorer工具中查看，依赖最底层的dll为jvm.dll。



4、使用Virbox Potector工具对jvm.dll进行加壳保护。



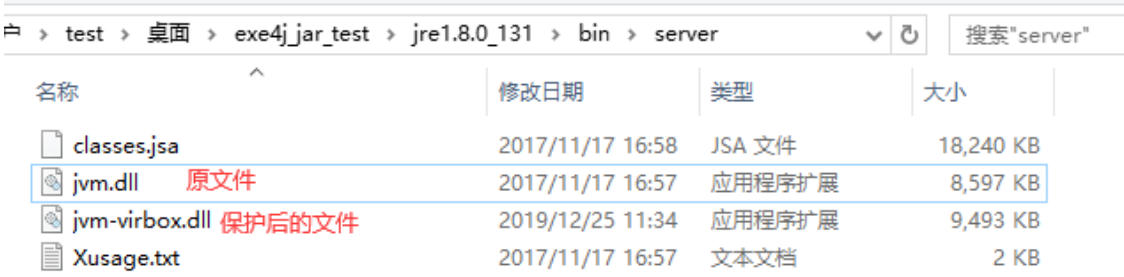
5、使用DSProtector工具对jar包进行加密。



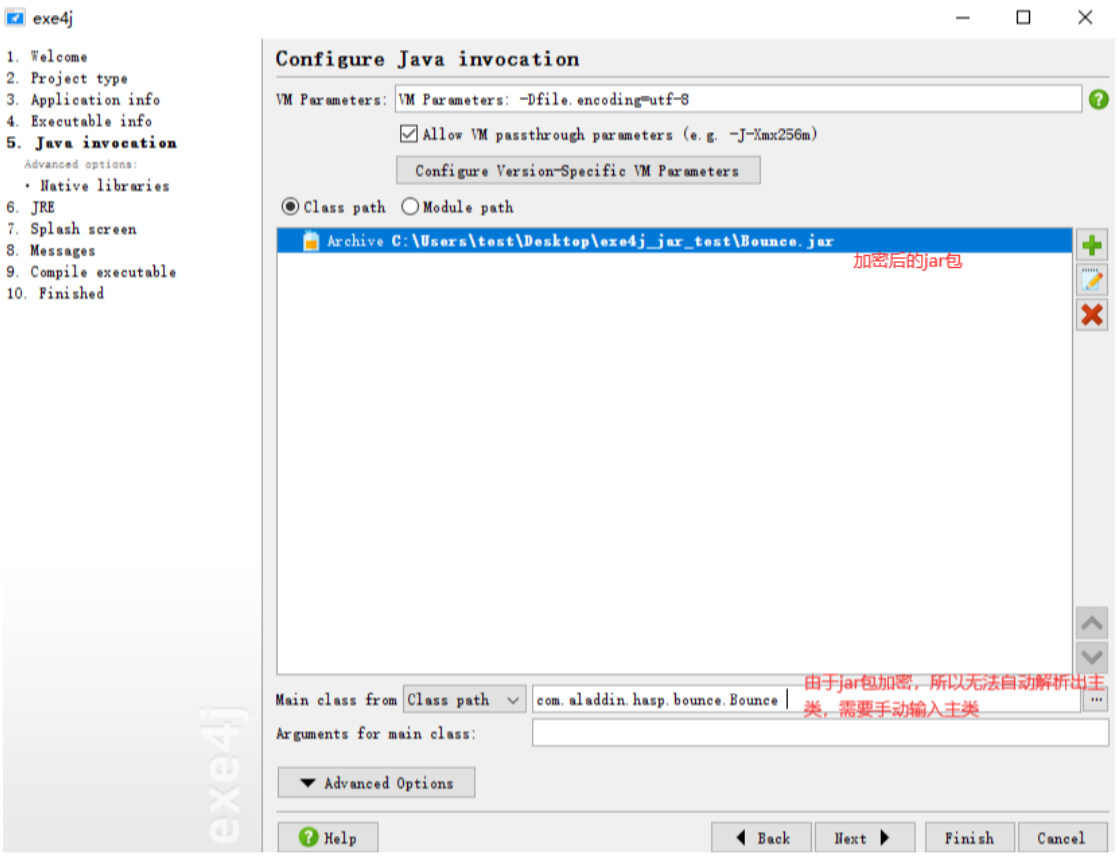
6、jar包加密成功后，使用exe4j对jar包打包，exe4j打包流程不变，但需要注意以下几点。

- 首先将保护后的主程序名称进行更改，如`.\jre1.8.0_131\bin\server\jvm-virbox.dll`，保证不与原文件名称重名，然后将原文件jvm.dll先放回原位置。

  【注意】：以上jvm.dll为测试环境路径，非固定路径，请根据自己环境进行查找。



- 使用exe4j工具打包，其中导入的jar包为加密后的jar包，主类需要手动输入。



- 打包时需要修改依赖java的环境，添加自己的java环境。

- 打包exe成功后，将`.\jre1.8.0_131\bin\server\jvm-virbox.dll`修改为`.\jre1.8.0_131\bin\server\jvm.dll`（要与原文件名称相同），否则程序无法正常运行。
- 运行exe时，使用procmon.exe依然可以监测到产生临时文件，但此时生成的临时文件中jar包是加密过后的。
- 最后，发布exe时需要将原文件的jvm.dll和备份的jar删除，避免泄露原始文件。

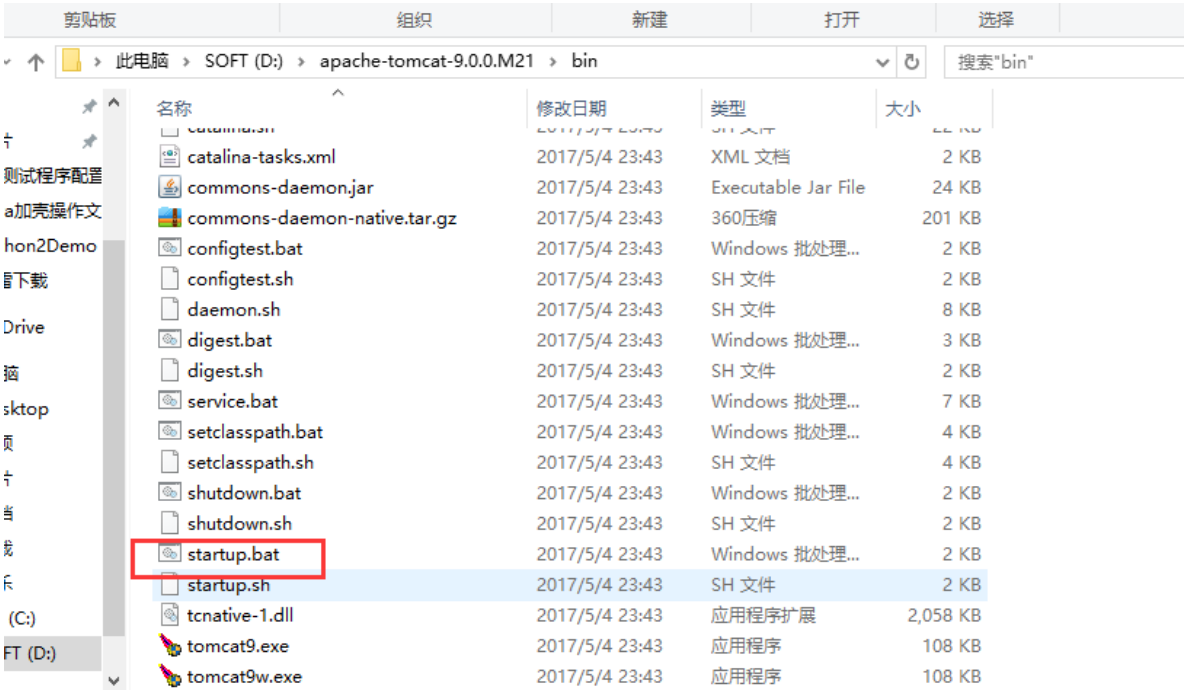
war文件保护

Windows

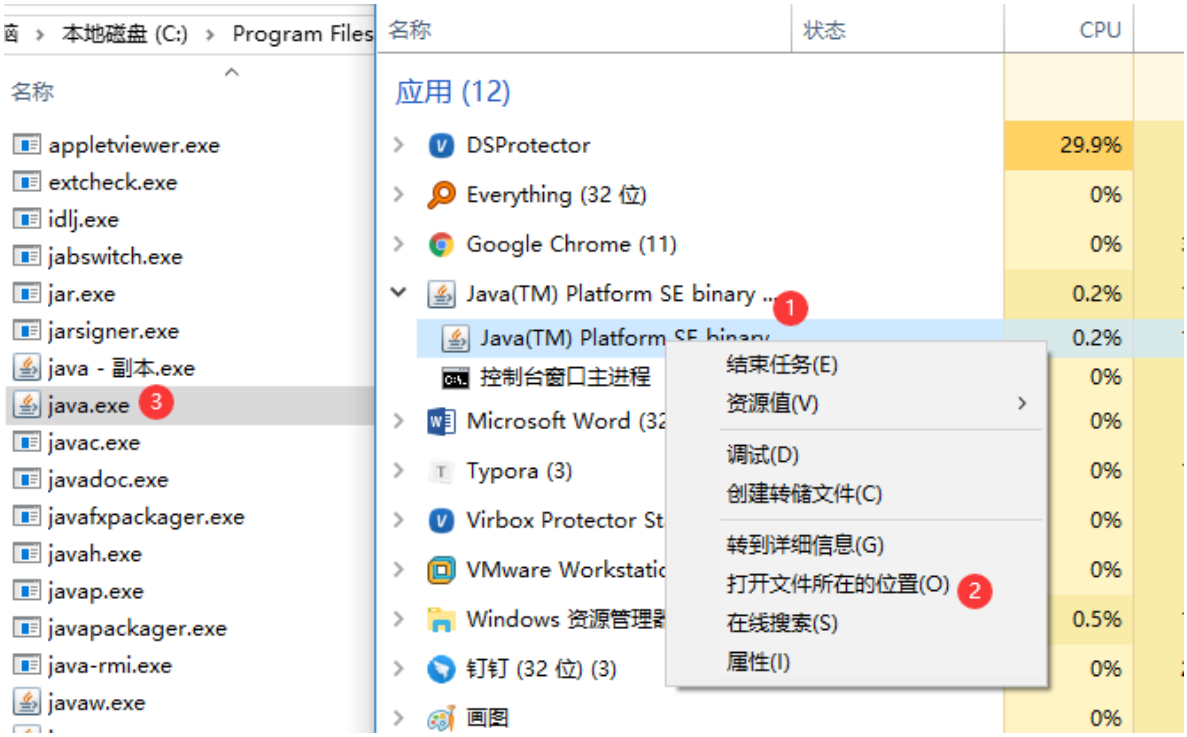
使用Virbox Protector工具对服务进程加壳保护，使用DSProtector对war包中的资源文件进行加密，以下介绍几种使用较多的场景操作流程。

startup.bat启动服务

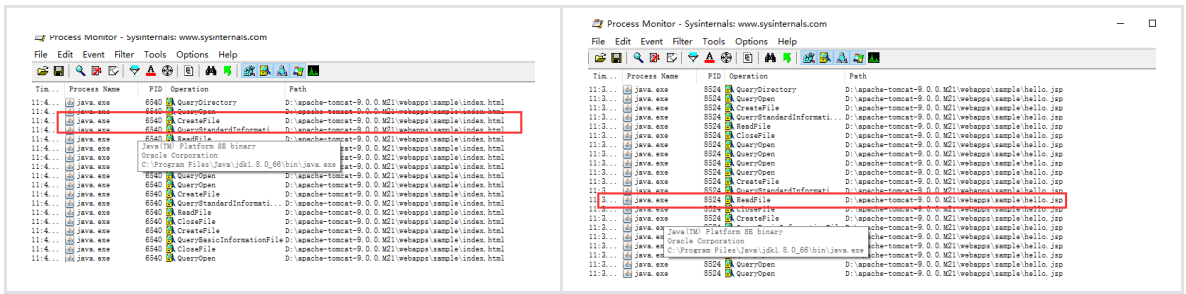
1、通过启动`.\apache-tomcat-9.0.0.M21\bin\startup.bat`来打开Tomcat服务。



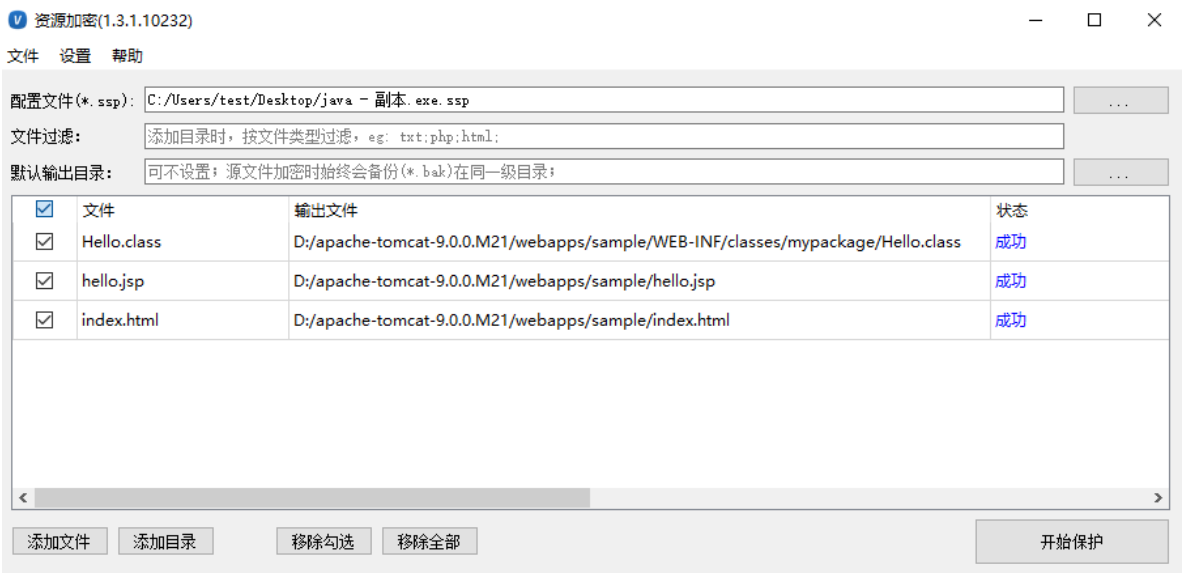
2、打开任务管理器，可以看到startup.bat启动时调用的java.exe进程。



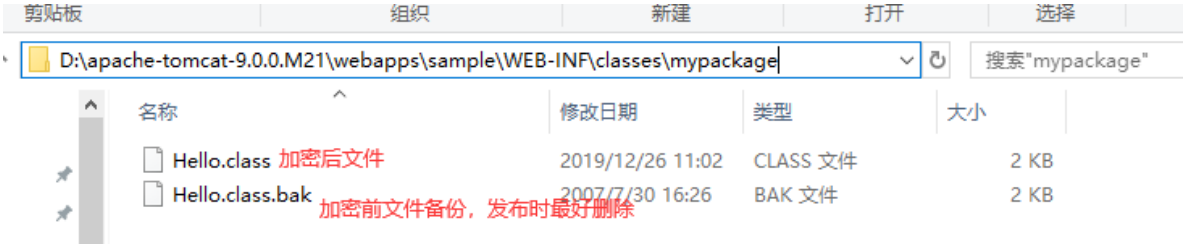
3、使用Procmon工具对war包的资源文件比如jsp、html、class文件进行监测，发现资源文件使用时被调用的主进程是为java.exe，这说明主进程找对了。



4、找到java.exe进程的位置，使用Virbox Protector工具对其加壳保护，选择要保护的war包class文件，使用DSProtector工具进行加密。



4、若未指定加密文件的输出路径，则文件加密成功后，原始文件会进行备份，发布时最好将其删除以防文件泄露。

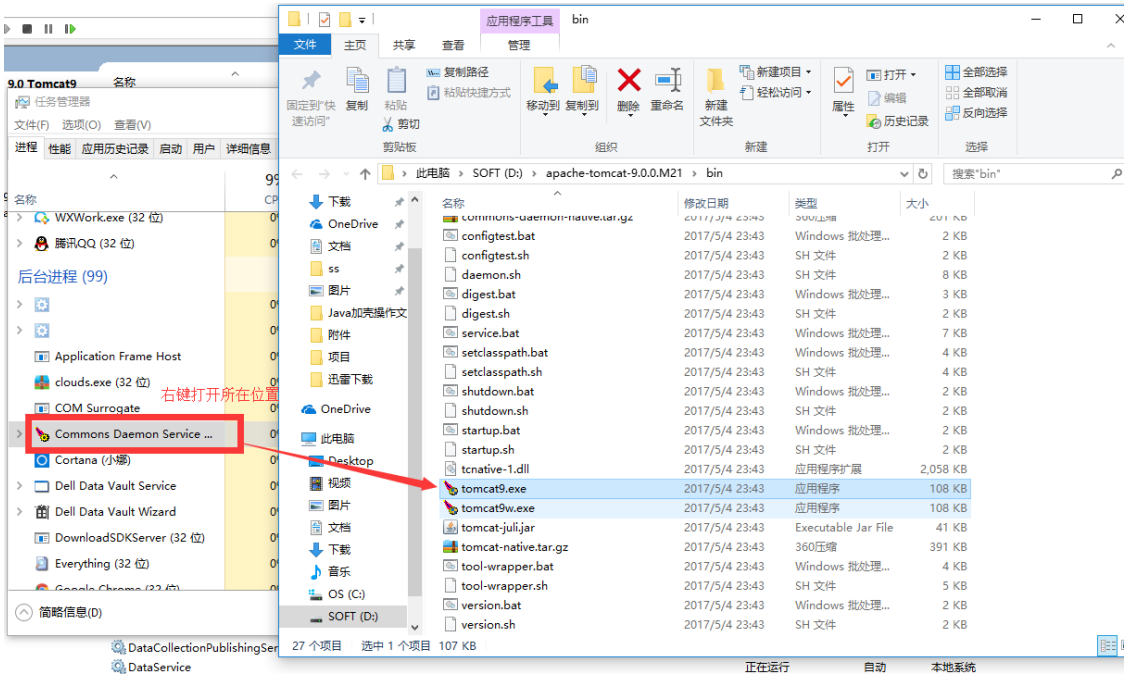


5、再次启动startup.bat服务，网站正常运行。

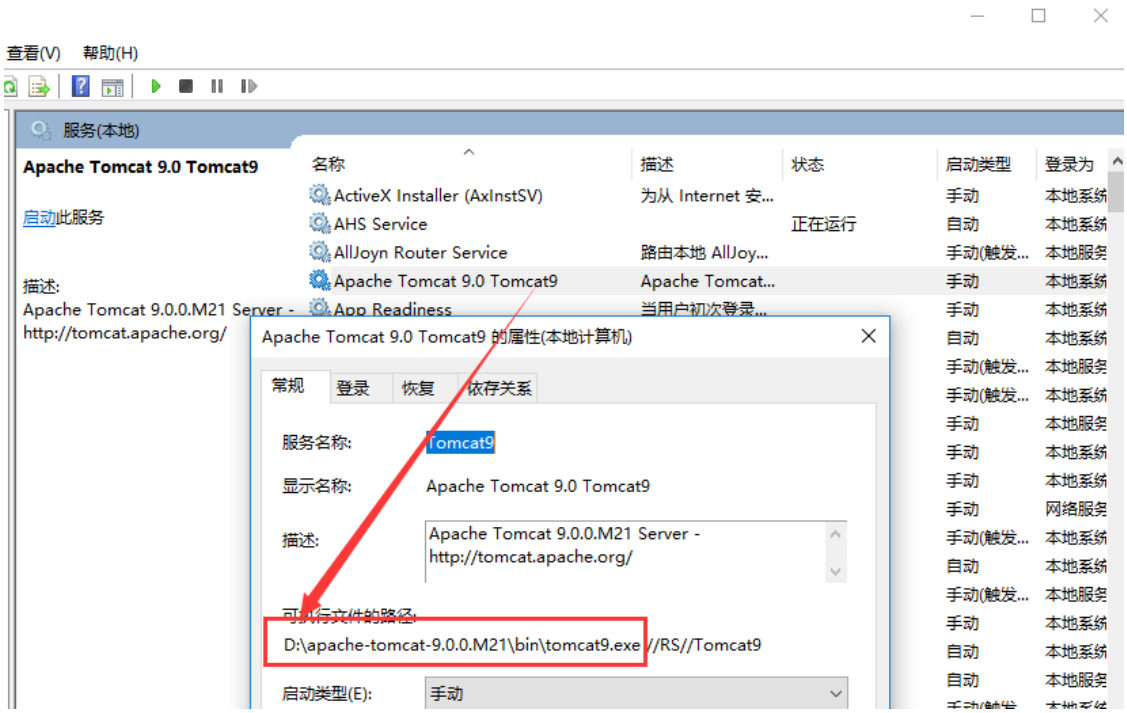
tomcat.exe启动服务

以Tomcat9为例，以下两种启动方式最终所依赖的进程为tomcat9.exe。

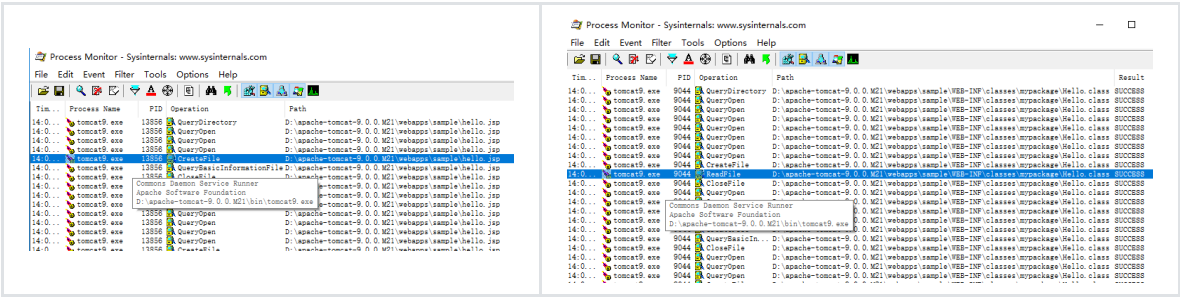
- 通过 `.\apache-tomcat-9.0.0.M21\bin\tomcat9.exe` 直接启动服务，如图：



- 通过 控制面板-管理工具-服务-`Apache Tomcat 9.0 Tomcat9` 启动服务，如图：



1、使用Procmon工具对war包的资源文件比如jsp、html、class文件进行监测，发现资源文件使用时被调用的主进程是为tomcat9.exe。



2、使用Virbox Protector工具对tomcat9.exe进行加壳，使用DSProtector对war包中的资源文件进行加密。

3、正常启动服务运行即可。

tomcat服务开机自启

场景：在windows系统上，使用**Virbox Protector LM体系工具**对Tomcat进程进行加壳保护，使用DSProtector对war包中资源文件文件进行加密后，使Tomcat设置成开机自启动后，网站能正常运行。

【注意】若使用**Virbox Protector Standalone体系工具**对tomcat进程进行加壳保护，不需要配置以下环境。

- 优点：可以将保护后的程序与许可进行关联，更加安全。
- 缺点：依赖Virbox许可服务，环境配置复杂。

文件保护操作步骤

1、首先使用Virbox Protector LM工具对tomcat9.exe加壳保护，保护后的tomcat进程依赖virbox用户工具。

2、若服务中tomcat9设置成开机自启，需要对tomcat和Virbox用户工具的服务设置依赖关系，若tomcat9的优先级高于SS服务，则会导致tomcat9自启动无法启动。

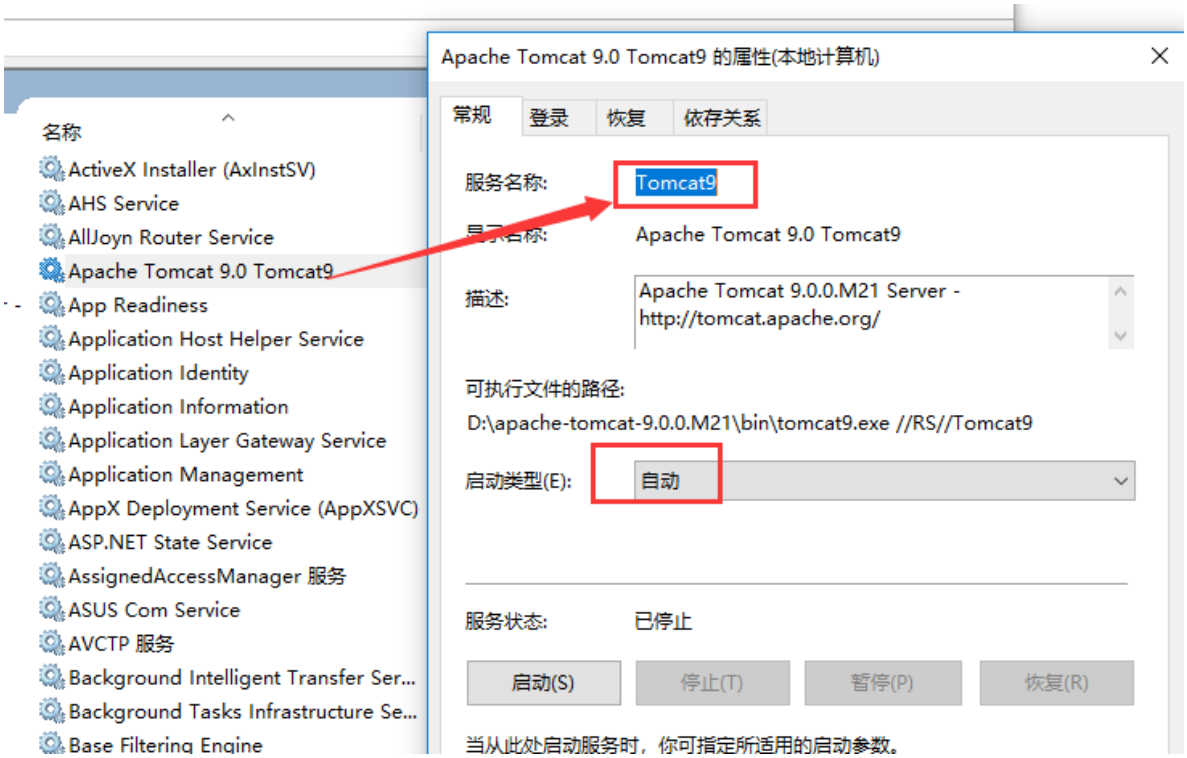
3、使用DSProtector对war包中的资源文件进行加密。

4、重启机器时，服务和网站都能正常运行。

设置服务依赖关系

方案一

1、进入控制面板-管理工具-服务页面，找到tomcat的服务名称。



2、将tomcat9服务启动方式设置为自动。

> 命令：sc config Tomcat9 start=auto

```
C:\Windows\system32>sc config Tomcat9 start=auto
[SC] ChangeServiceConfig 成功
```

3、设置tomcat服务在sense shield服务启动后再启动。

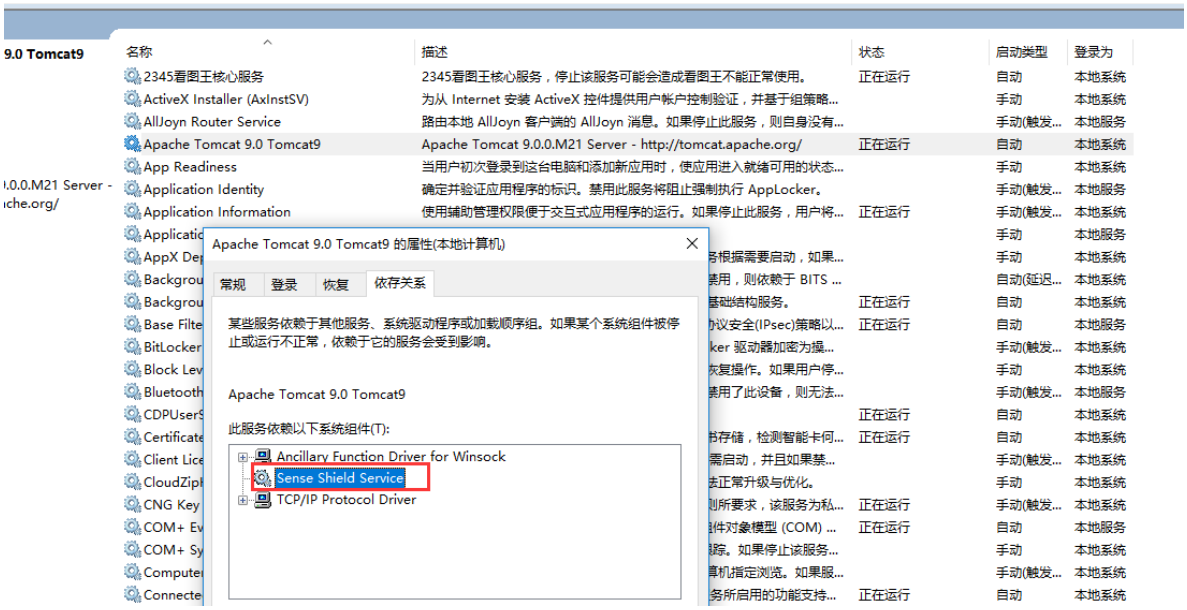
> 命令：sc config Tomcat9 depend= "Sense Shield Service"

```
C:\Windows\system32>sc config Tomcat9 depend= "Sense Shield Service"
[SC] ChangeServiceConfig 成功
```

- 如果想要把服务依赖关系移除，可以使用下面的命令。

> 命令：sc config Tomcat9 depend= ""

4、在`控制面板-管理工具-服务-Apache Tomcat 9.0 Tomcat9-鼠标右键-属性`，查看设置的依存关系已生效。

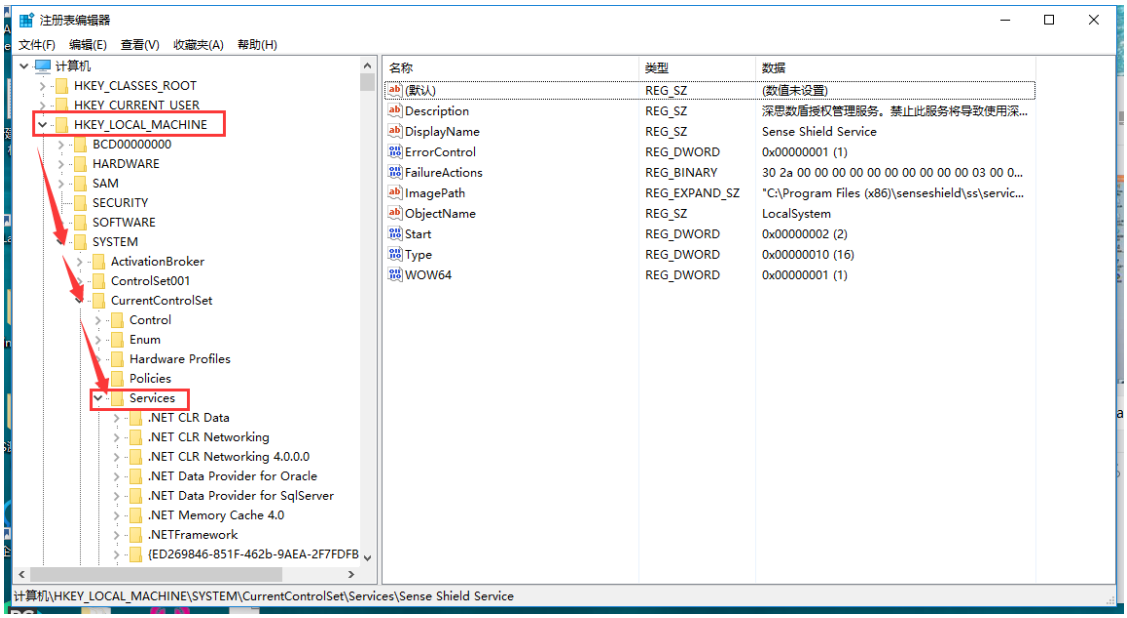


方案二

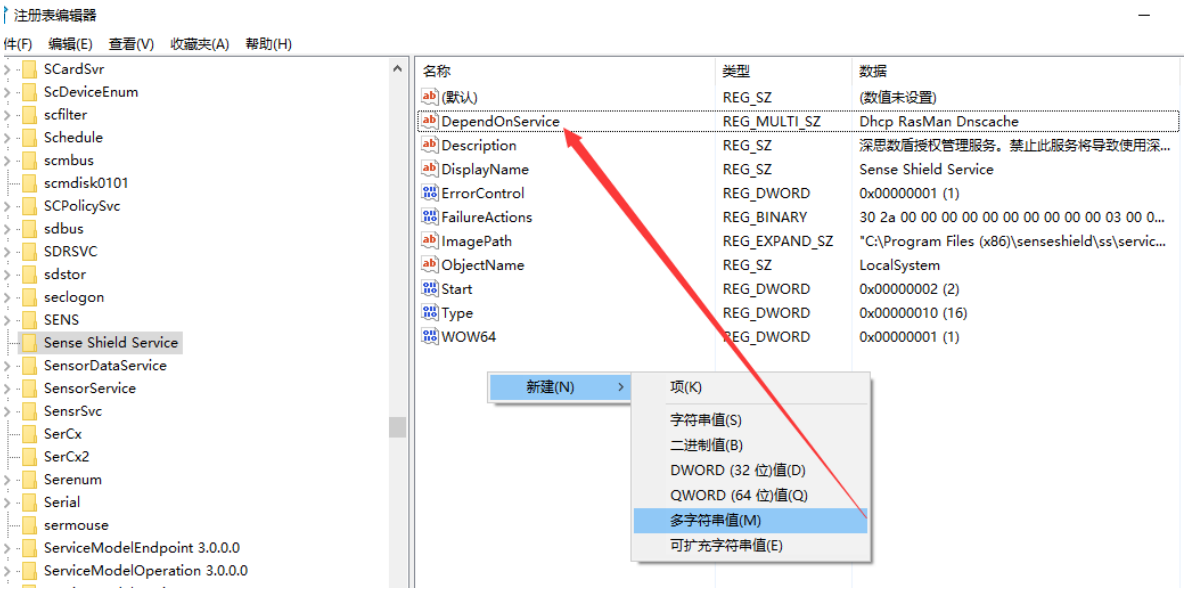
【注意】使用这种设置依赖方式，如果SS服务重新安装，服务中设置的依赖关系会被覆盖，需要进行重新设置。

若使用云锁，ss服务中云锁登陆需要依赖网络的启动，所以需要设置SS服务和网络服务的依赖关系，重新启动时，云锁才能登陆成功。

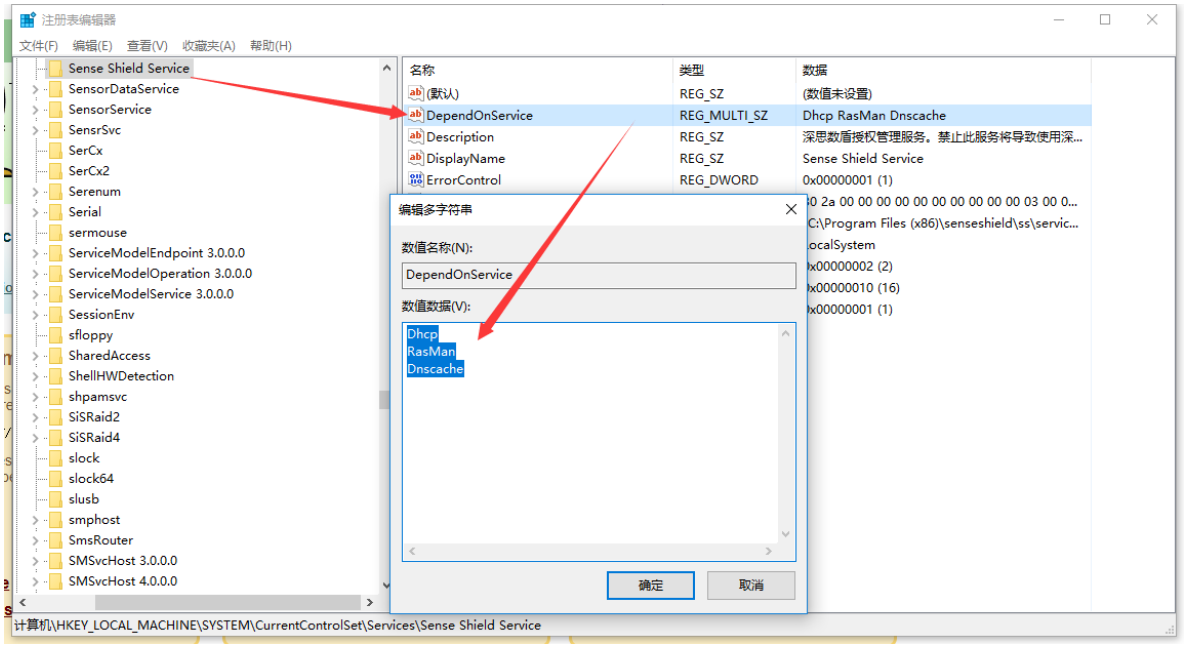
- 输入regedit进入注册表编辑器，找到Sense Shield Service服务。如图所示：



- 若没有DependOnService这个字符串，需要在空白处右键新建多字符串值，将名字改为DependOnService。若存在DependOnService字符串，则直接进行下一步。如图所示：

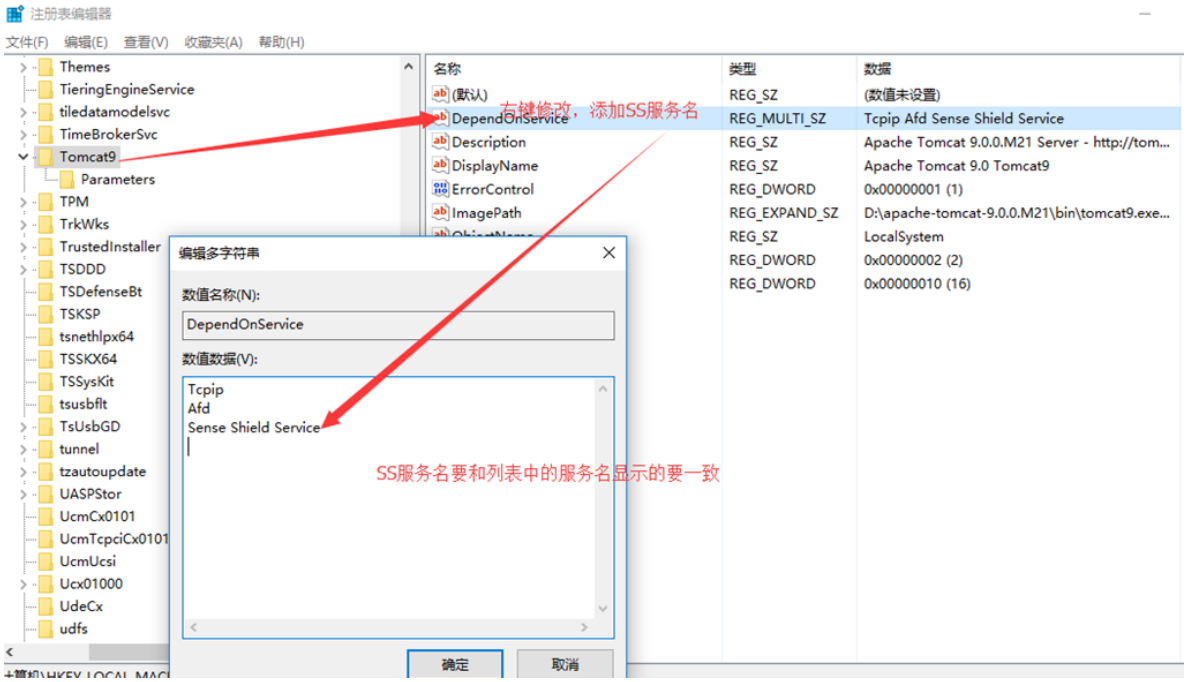


- 点击DependOnService字符串，右键-修改，输入Dhcp、RasMan、Dnscache三个数值，点击确定。如图所示：

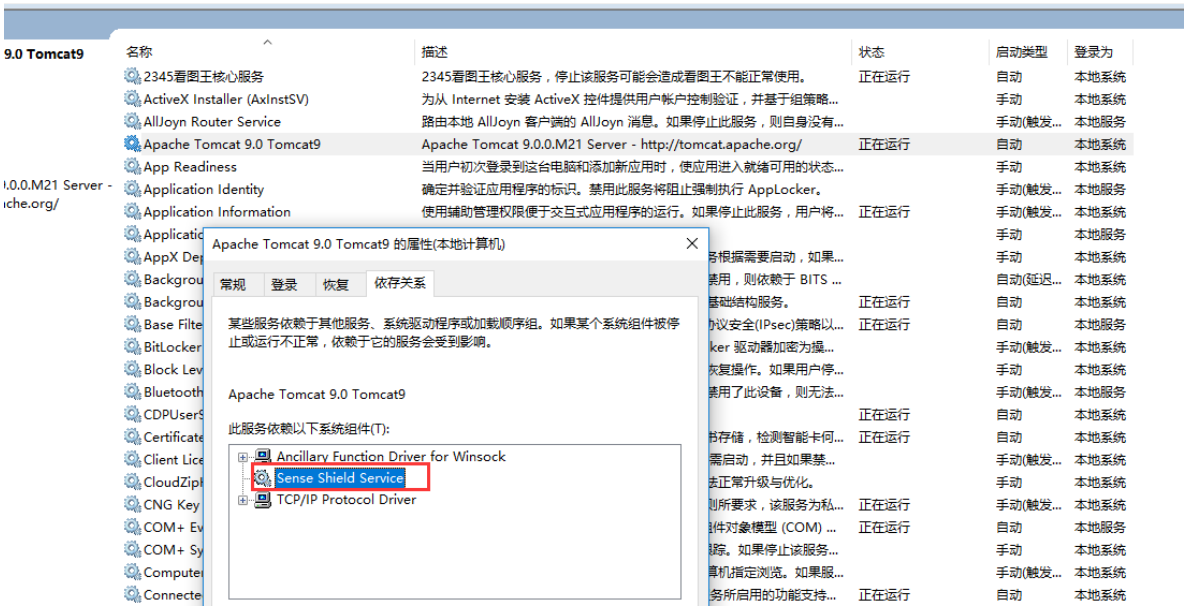


- 在HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tomcat9找到tomcat9文件夹，右键-修改DependOnService字符串的数值，添加Sense Shield Service服务名称。如图所示：

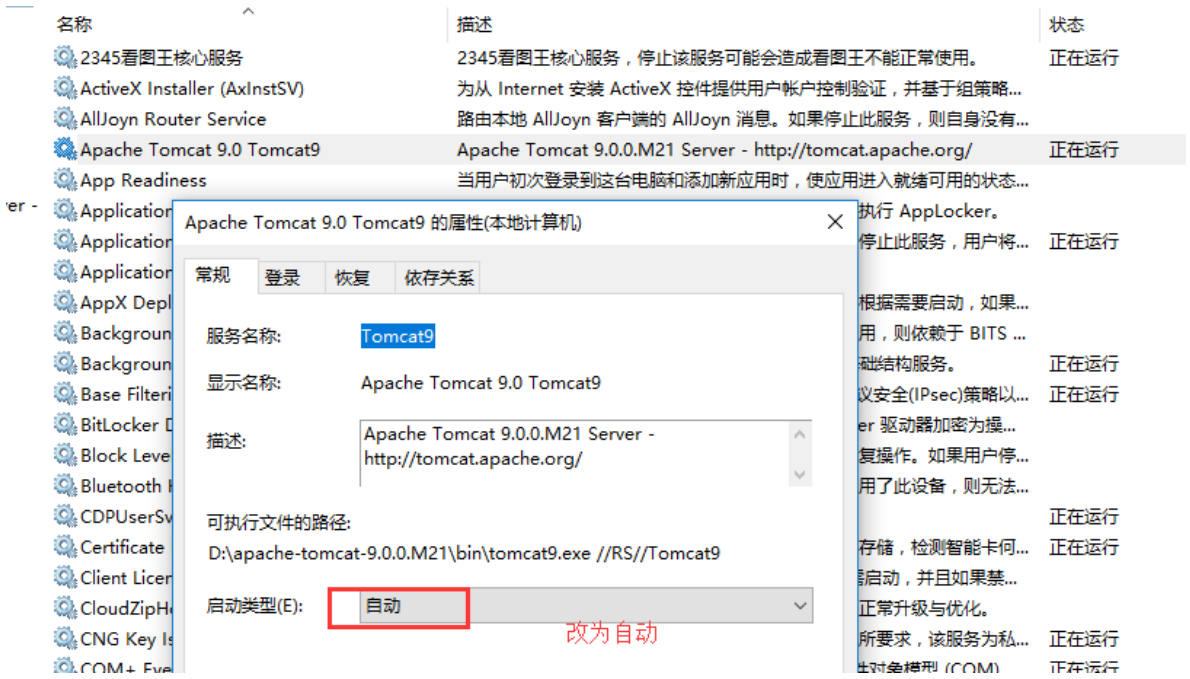
  【注意】添加数值数据，添加的数据最后一栏不要有空格，否则将提示错误。



- 在 控制面板-管理工具-服务-Apache Tomcat 9.0 Tomcat9-鼠标右键-属性，查看设置的依存关系已生效。



- 在 控制面板-管理工具-服务-Apache Tomcat 9.0 Tomcat9-鼠标右键-属性，点击常规选项，tomcat服务启动类型更改为自动。



Linux

tomcat服务

以centos7系统上tomcat9服务为例：

1、启动 `./apache-tomcat-9.0.30/bin/startup.sh` 服务，根据配置jdk环境变量不一样，JRE_HOME所指定的位置不同。

```
[root@localhost bin]# ps -ef | grep java
root       4552   3461  0 01:24 pts/0    00:00:00 grep --color=auto java
[root@localhost bin]# ./startup.sh
Using CATALINA_BASE:   /usr/local/tomcat/apache-tomcat-9.0.30
Using CATALINA_HOME:   /usr/local/tomcat/apache-tomcat-9.0.30
Using CATALINA_TMPDIR: /usr/local/tomcat/apache-tomcat-9.0.30/temp
Using JRE_HOME:        /usr/local/java/jdk1.8.0_221
Using CLASSPATH:       /usr/local/tomcat/apache-tomcat-9.0.30/bin/bootstrap.jar:
/usr/local/tomcat/apache-tomcat-9.0.30/bin/tomcat-juli.jar
Tomcat started.
```

2、tomcat服务中依赖jdk环境，启动tomcat服务调用进程是java，使用命令可以查看到进程java的位置。

```
[root@localhost bin]# ps -ef | grep java
root       4585      1 21 01:25 pts/0    00:00:02 /usr/local/java/jdk1.8.0_221/b
in/java -Djava.util.logging.config.file=/usr/local/tomcat/apache-tomcat-9.0.30/c
onf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLo
gManager -Djdk.tls.ephemeralDHKeySize=2048 -Djava.protocol.handler.pkgs=org.apac
he.catalina.webresources -Dorg.apache.catalina.security.SecurityListener.UMASK=0
027 -Dignore.endorsed.dirs= -classpath /usr/local/tomcat/apache-tomcat-9.0.30/bi
n/bootstrap.jar:/usr/local/tomcat/apache-tomcat-9.0.30/bin/tomcat-juli.jar -Dcat
alina.base=/usr/local/tomcat/apache-tomcat-9.0.30 -Dcatalina.home=/usr/local/tom
cat/apache-tomcat-9.0.30 -Djava.io.tmpdir=/usr/local/tomcat/apache-tomcat-9.0.30
/temp org.apache.catalina.startup.Bootstrap start
root       4640   3461  0 01:25 pts/0    00:00:00 grep --color=auto java
[root@localhost bin]#
```

3、使用Virbox Protector工具对java.exe进行加壳，使用DSProtector对war包中的资源文件进行加密。

4、重新启动服务，程序正常运行即可。

python文件

支持范围

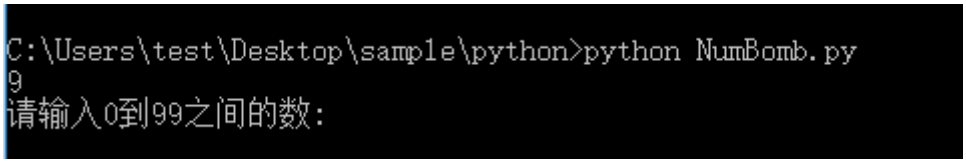
若不符合以下列表中场景的用户，可以联系**深思客服**详细咨询。

场景	是否支持
Anconda2/Anconda3	支持
C++/C语言文件中调用python	支持
py文件中调用C++/C	支持
pyinstall/py2exe将py文件打包成exe	支持
py文件转成的pyd/SO/可执行文件	支持

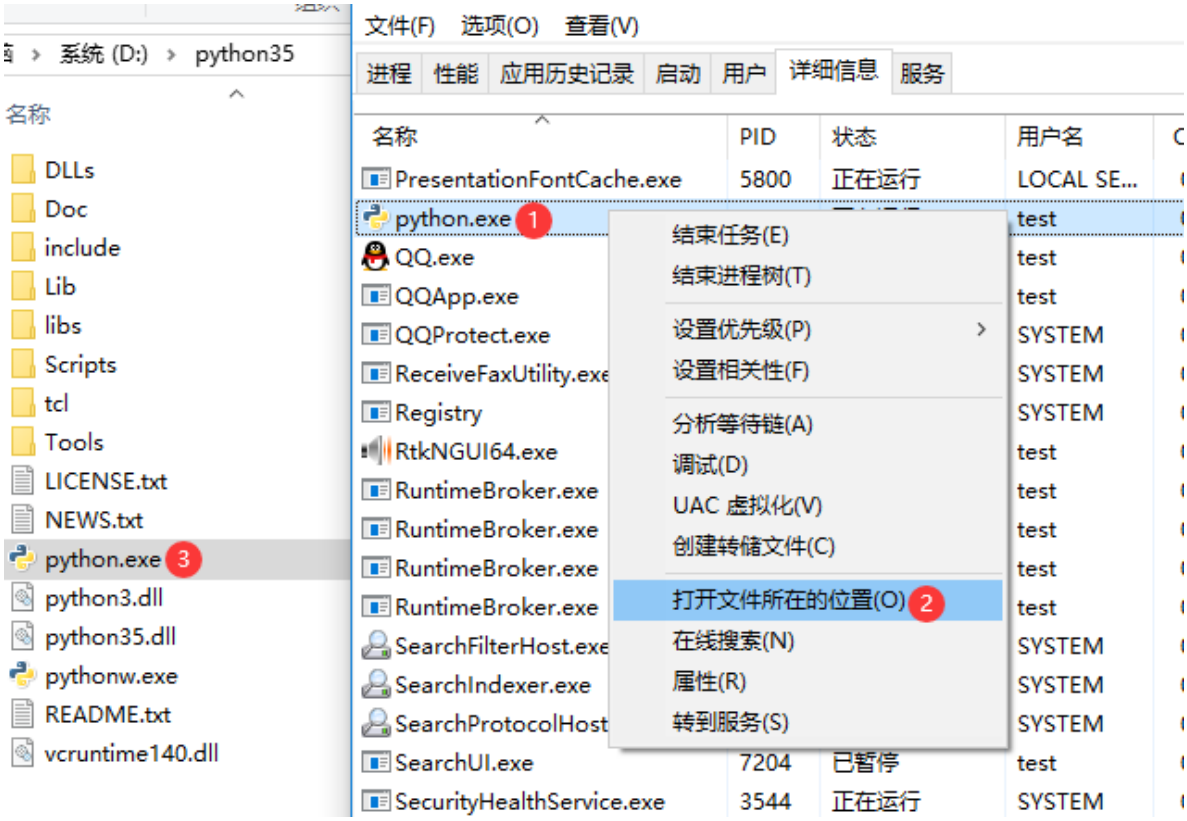
Windows

场景：直接命令行中运行python文件。

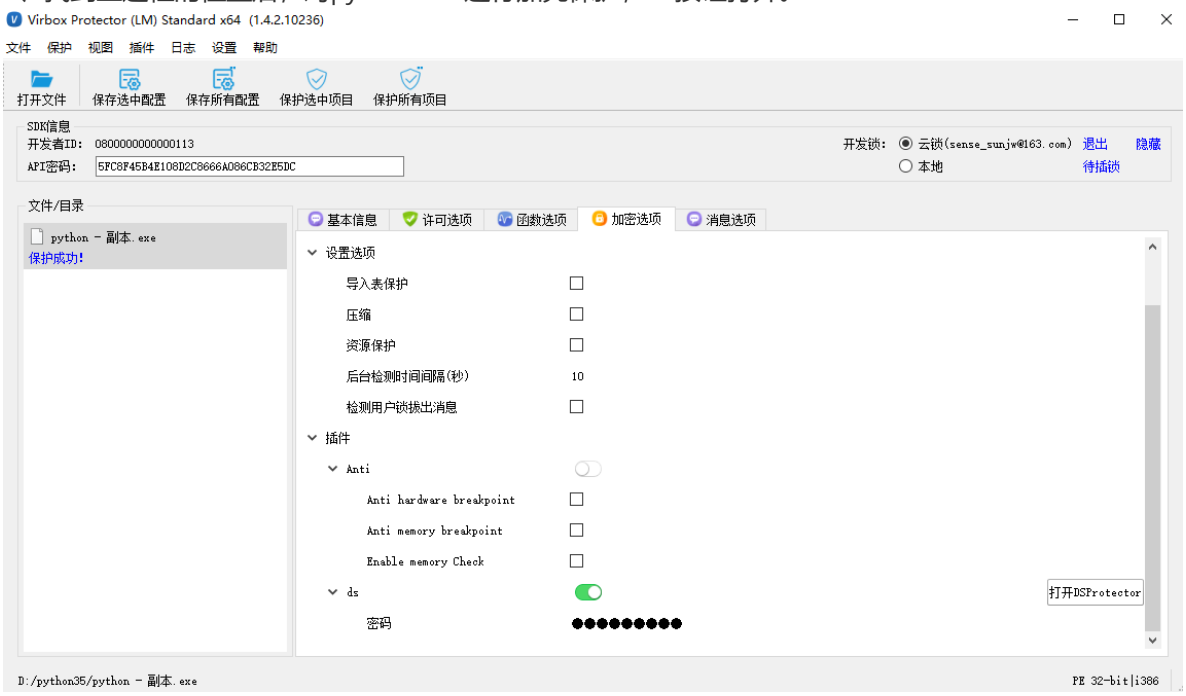
1、命令行直接执行py或pyc文件，比如：python demo.py或python demo.pyc



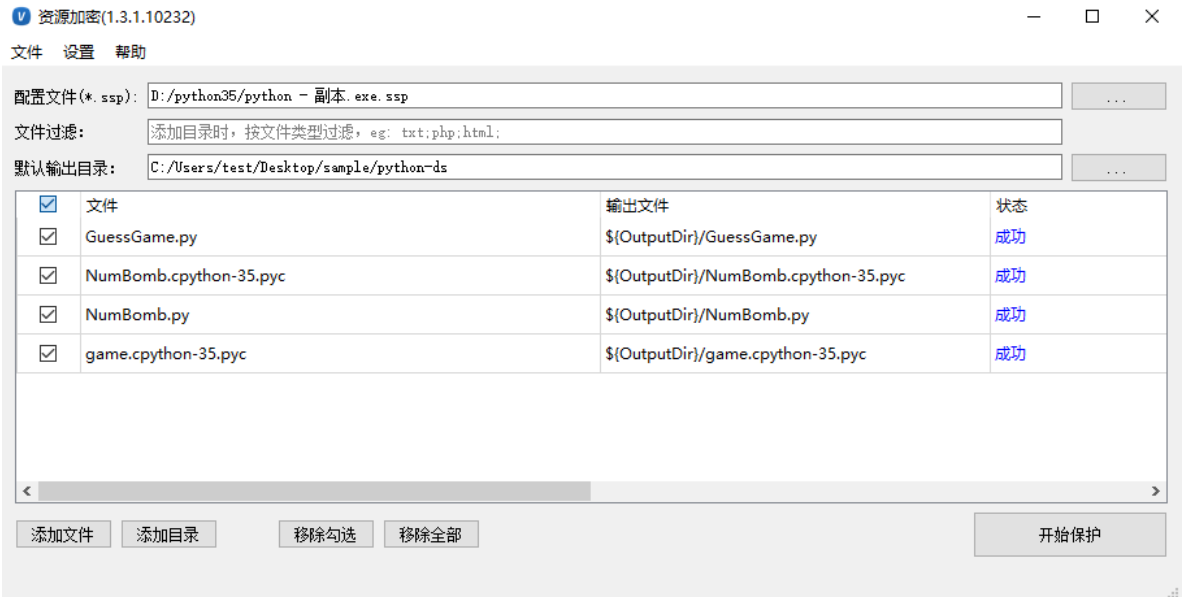
2、可在任务管理器中查找py/pyc文件调用的主进程。



3、找到主进程的位置后，对python.exe进行加壳保护，ds按钮打开。



4、对py或pyc文件进行加密。



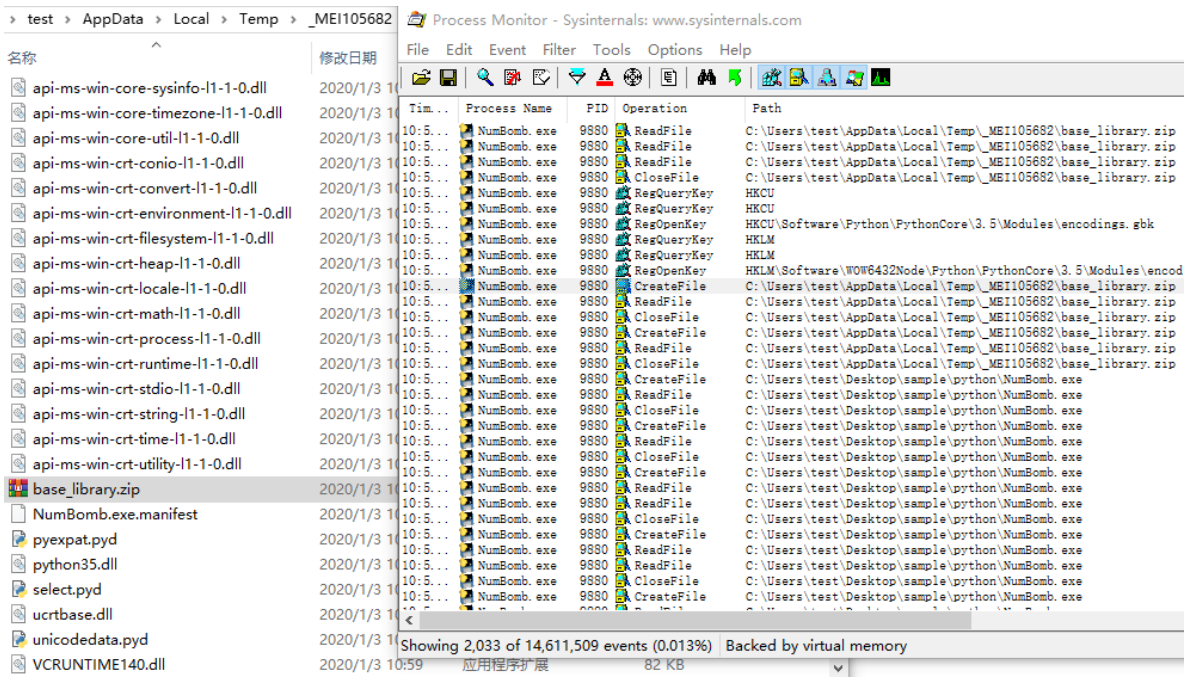
5、保护完毕后，可以正常运行。

pyinstall/py2exe打包exe

场景：在Windows系统上，使用python自带打包工具pyinstall/py2exe将py文件打包成单个exe。应该怎样保护?

以python3.5/pyinstall为例，操作如下：

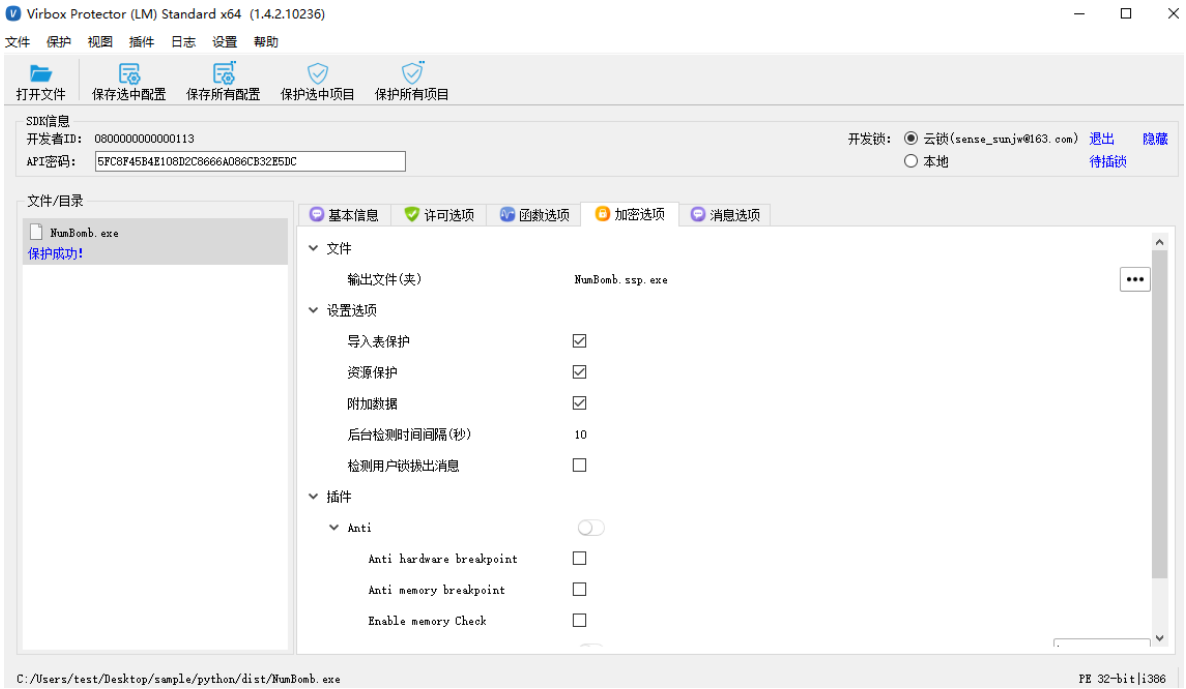
【注意】以下两种方案均不能避免exe运行产生临时文件的情况。



方案一

针对pyinstall或py2exe将py文件打包成的exe程序，可以直接使用Virbox Protect工具对exe文件进行加壳保护。

- 优点：方便，快捷。
- 缺点：由于exe运行会产生临时文件（Virbox Protect工具只对exe进行保护并不会改变exe的功能），可以通过监测exe找到缓存文件的位置，获取未加密文件。

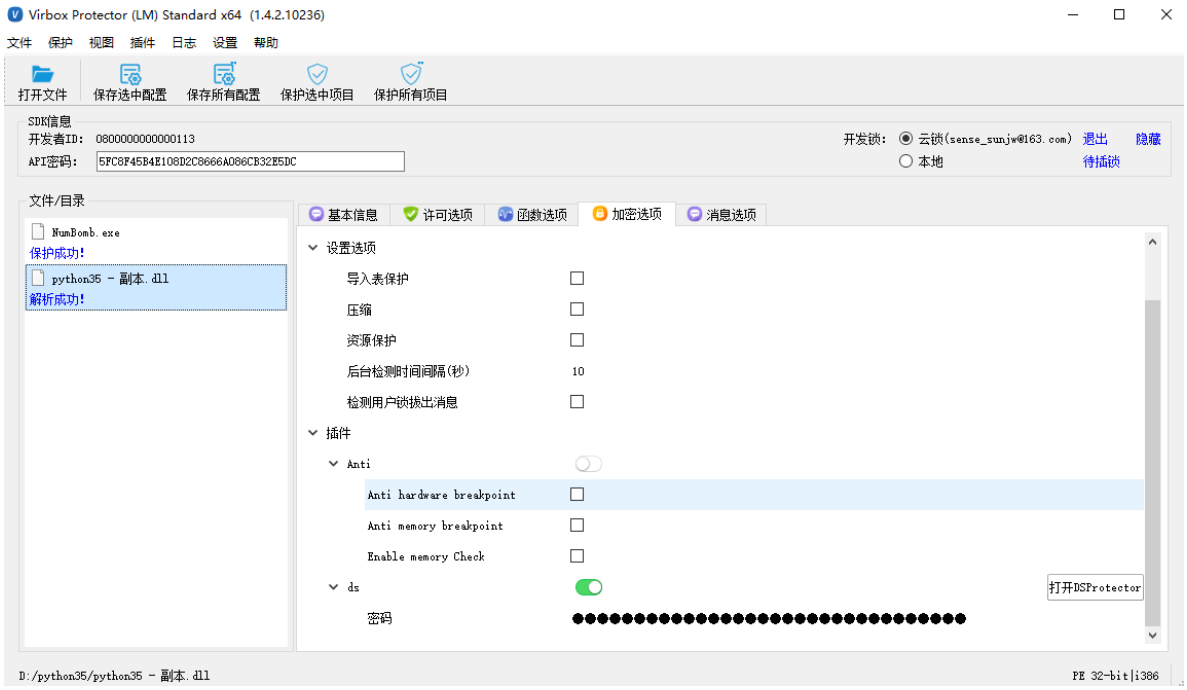


方案二

pyinstall打包时会将python35.dll库打包到exe中，且python.exe依赖python35.dll。

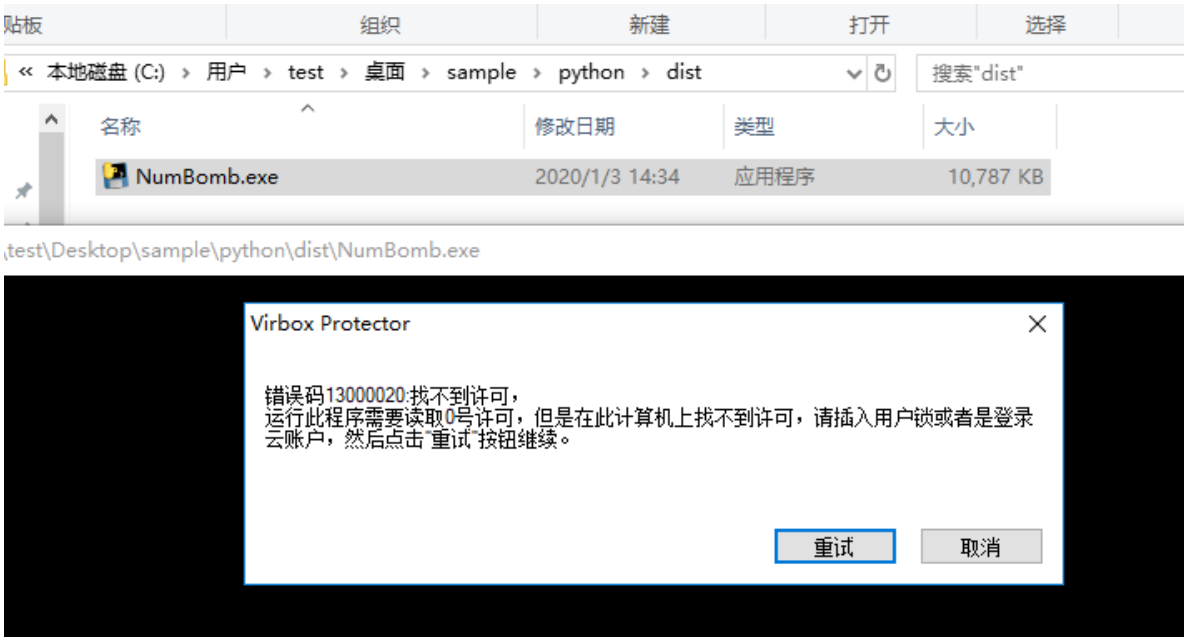
- 优点：产生的临时文件中的python35.dll是保护过后的，相对方案一较安全。
- 缺点：操作流程相对方案一较复杂。

1、使用Virbox Protector工具对python35.dll进行加壳保护。



2、DSProtector对py/pyc文件进行加密，加密操作完成后，使用pyinstall打包exe。

- 若对python35.dll是使用**Virbox Protector LM工具**加壳保护的，则打包后的exe直接运行将依赖许可，此时不需要对exe再次使用**Virbox Protector LM工具**对exe进行加壳。如图所示，



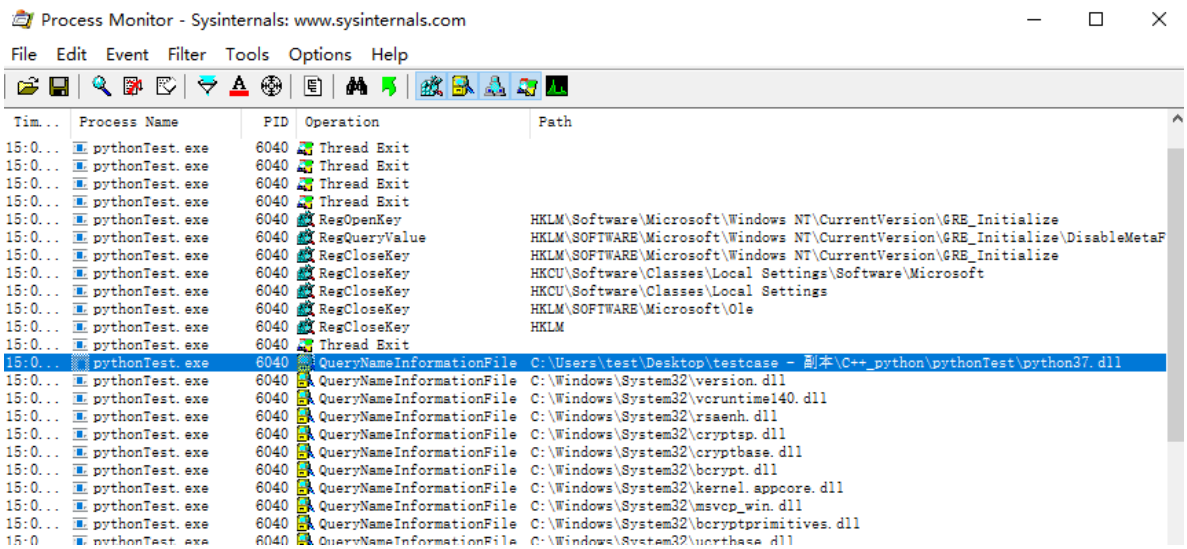
- 若对python35.dll是使用**Virbox Protector Standalone工具**加壳保护的，则打包后的exe不会依赖许可。

C++文件中调用python文件

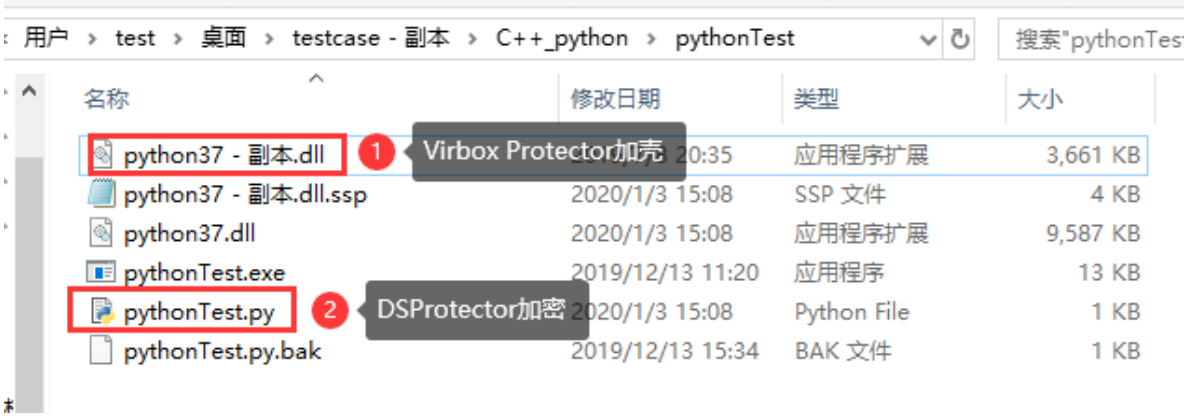
场景：在Windows系统上，使用C++语言的文件中调用py文件，将C++文件打包成exe后，应该如何保护?

以python3.7为例：

1、使用Procmon.exe查看打包后的exe运行所依赖的文件，可以看到是python37.dll。



2、然后使用Virbox Protector工具对python37.dll进行加壳，使用DSProtector对py文件进行加密，加密操作完成后，即可操作运行。

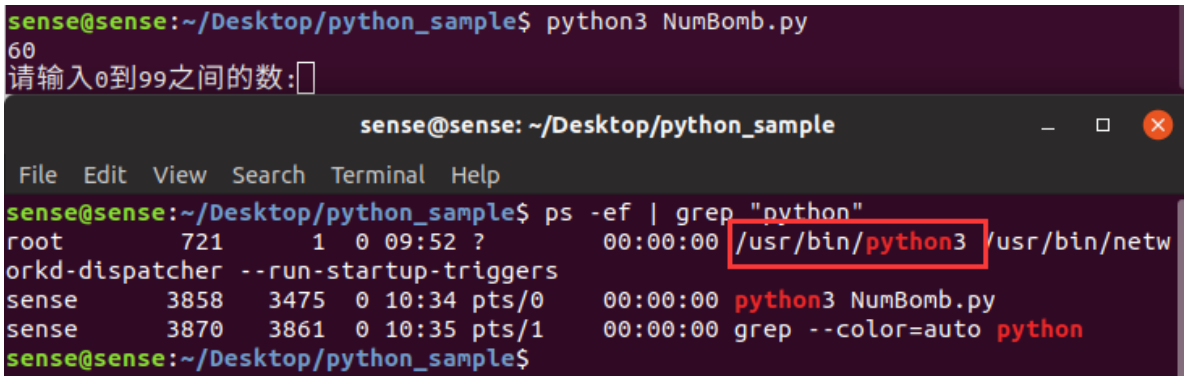


Linux

场景：直接命令行中运行python文件。

1、命令行直接执行py或pyc文件，比如：python demo.py或python demo.pyc。

2、查找py/pyc文件调用的主进程。



3、找到主进程后，再依次使用Virbox Protector工具对主进程加壳，DSProtector对py/pyc文件进行加密，加密操作完成后，程序可以正常运行。

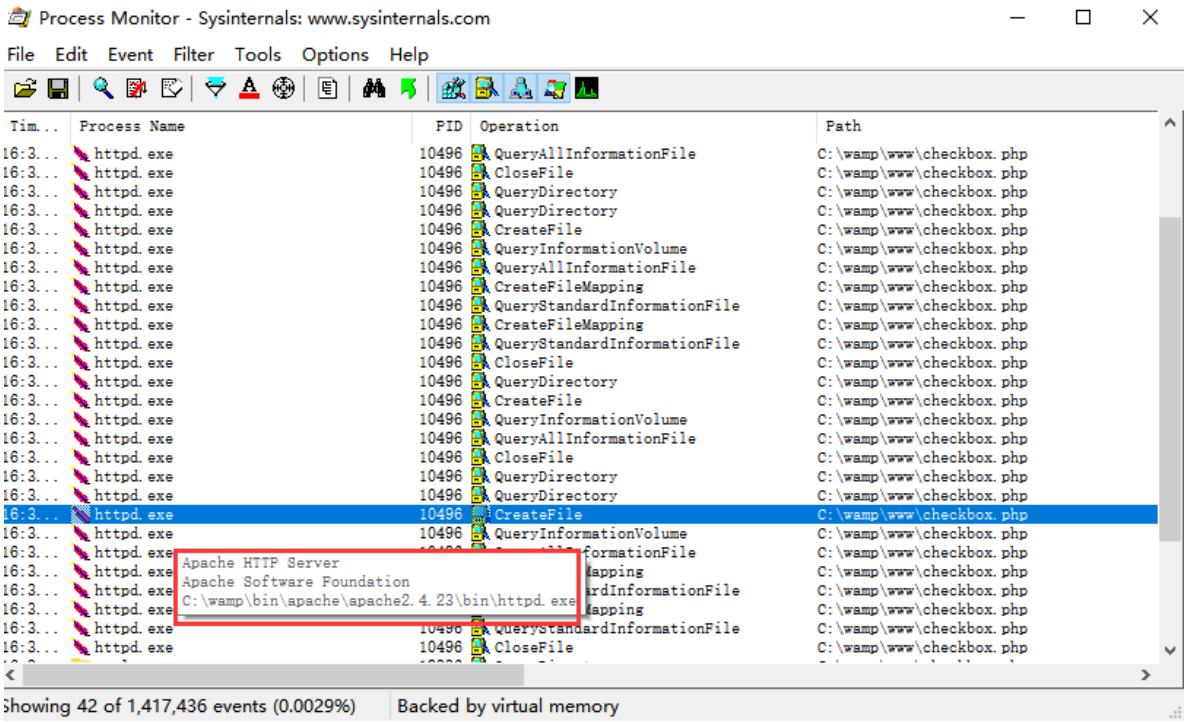
PHP文件

Windows

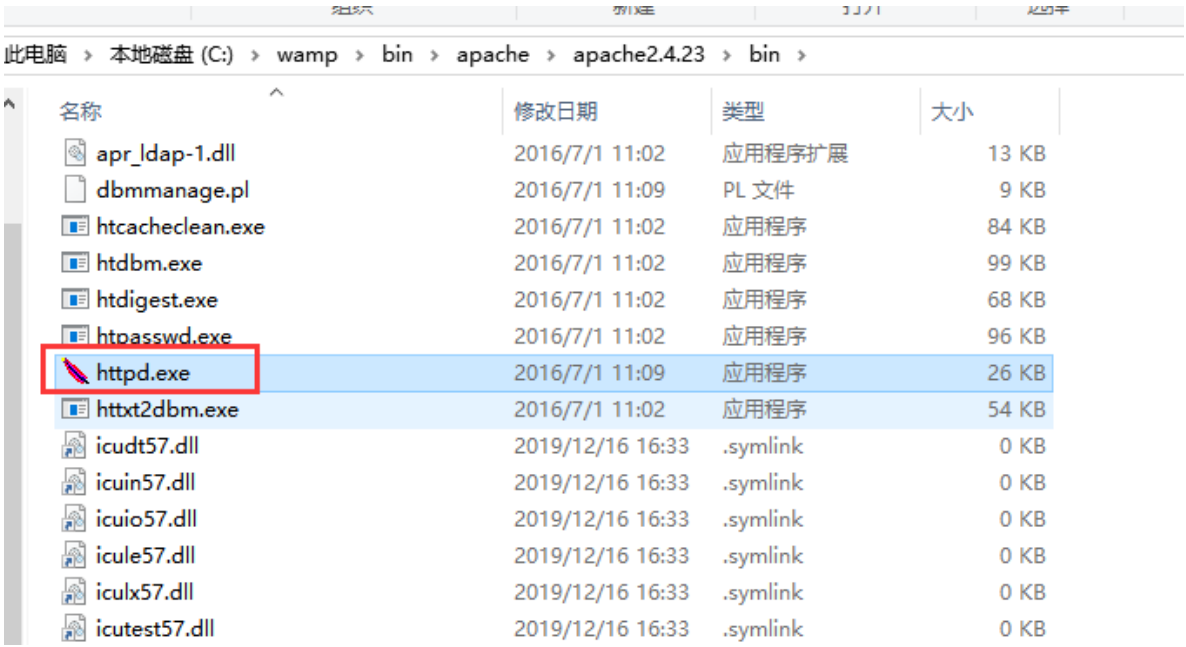
wampserver服务

场景：wampserver是Apache Web服务器、PHP解释器以及MySQL数据库的整合软件包，在Windows系统上使用wampserver软件包运行PHP程序，如何使用DSProtector加密呢?

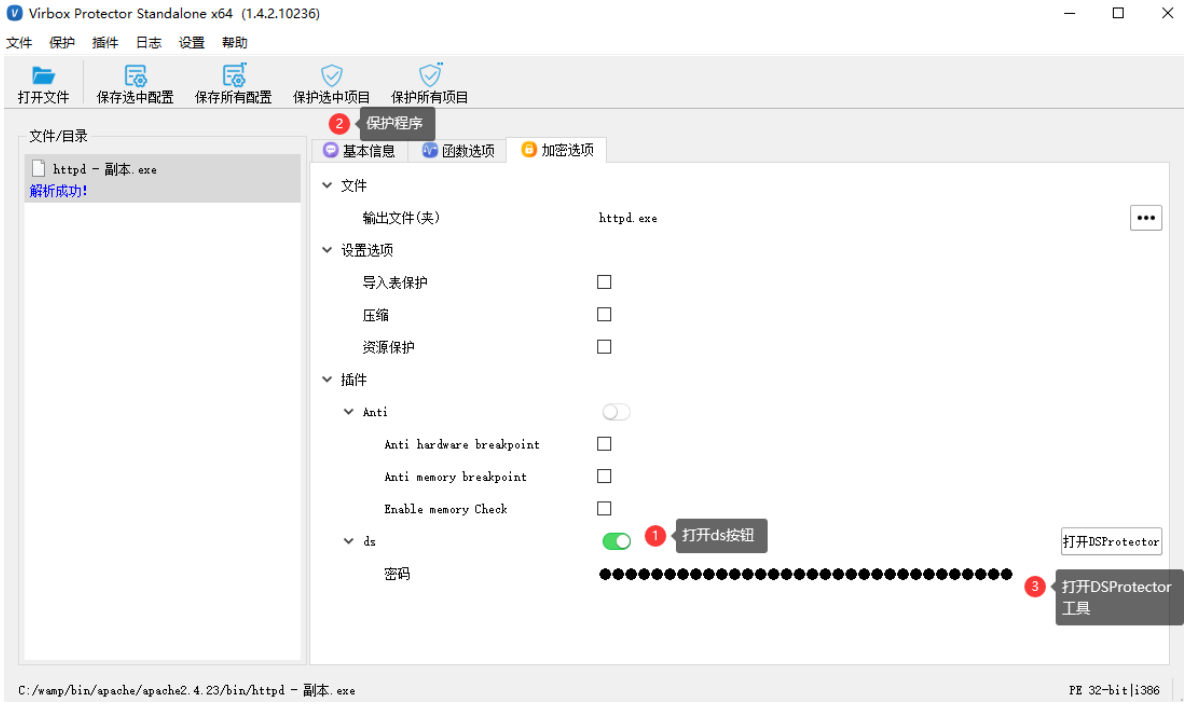
1、运行原始程序，借用Procmon监测工具监测调用 `checkbox.php` 的进程为 `httpd.exe`。



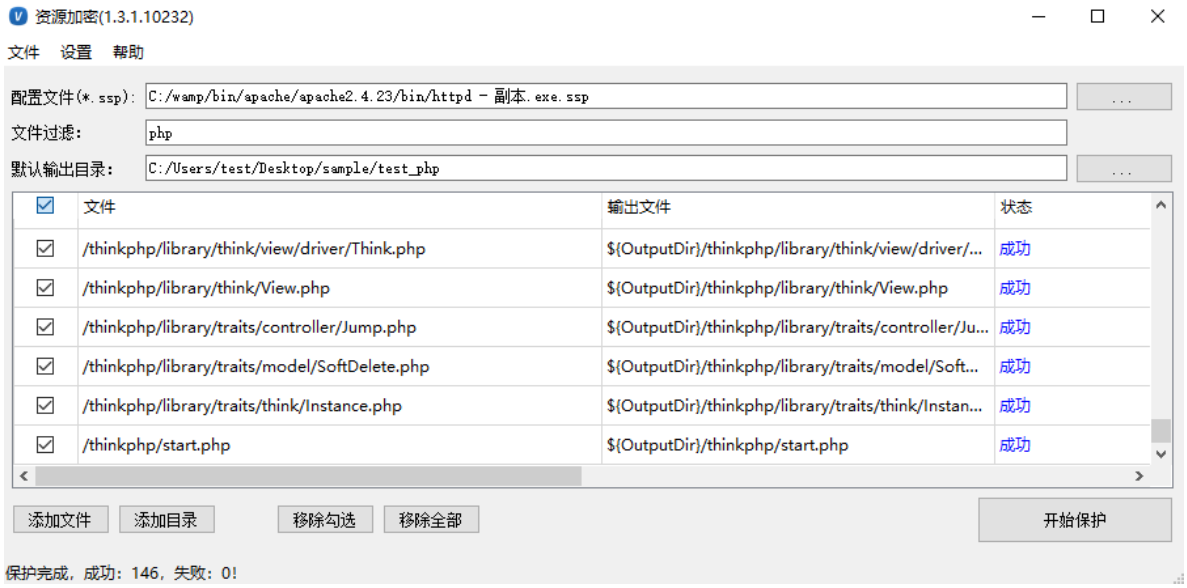
2、查到 `httpd.exe` 的位置。



3、对进程 `httpd.exe` 加壳保护。



5、对资源文件进行加密保护。

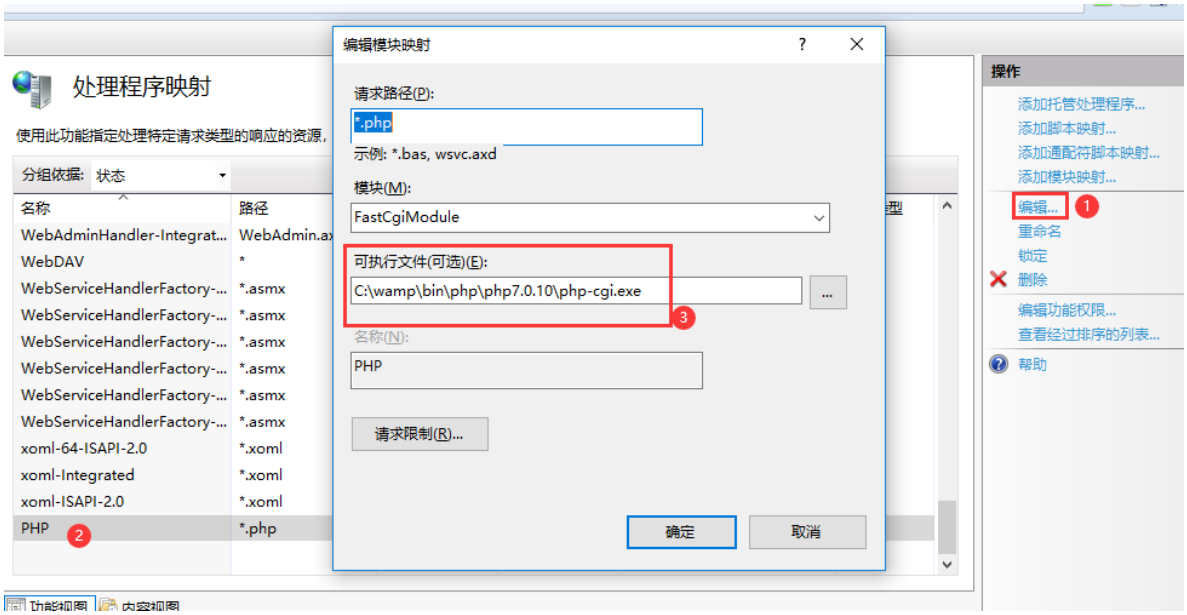


6、保护成功后，正常运行即可。

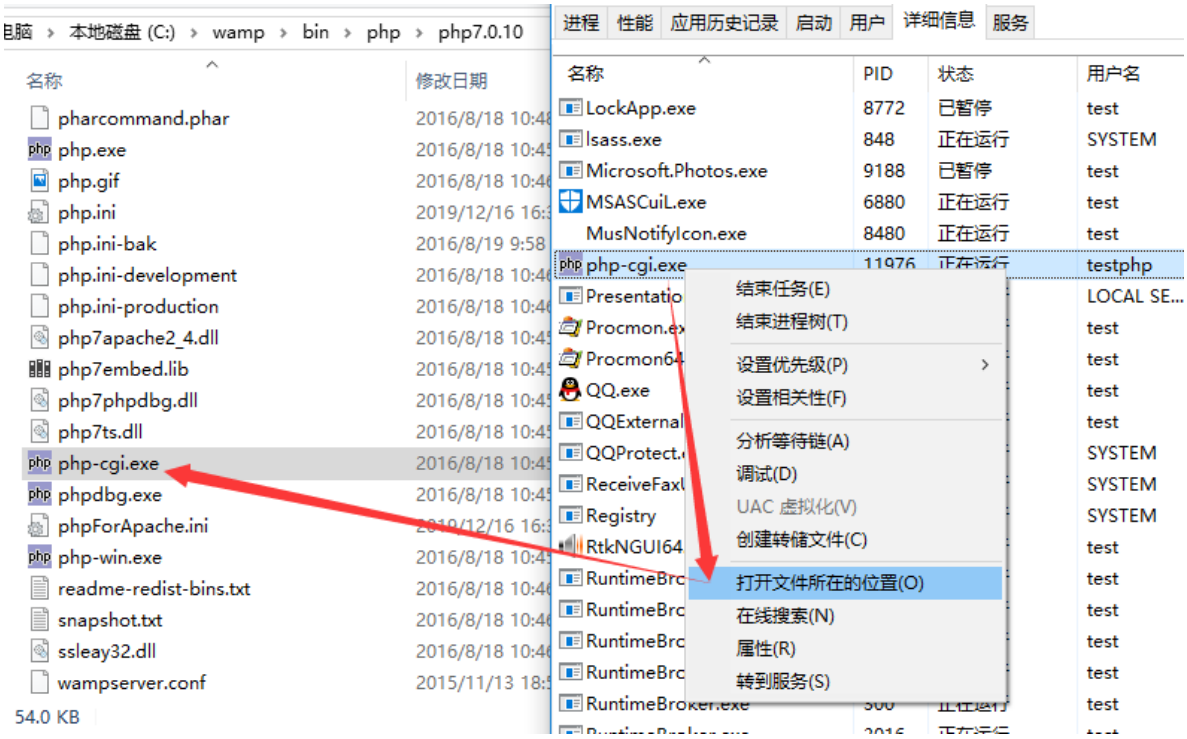
IIS服务

场景：在Windows系统上通过IIS服务启动PHP页面。

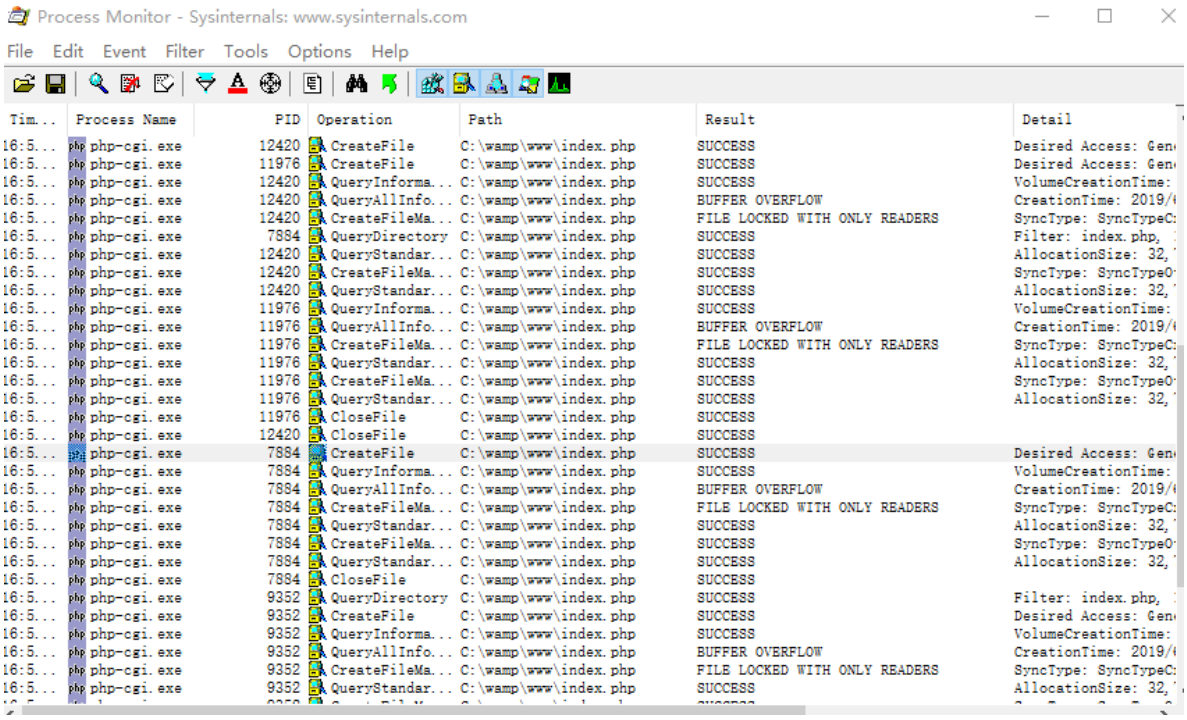
1、在 控制面板-管理工具-IIS 打开IIS服务页面，可以看到调用PHP主进程的是php-cgi.exe。



2、启动服务后，可以在任务管理器中进程并找到其位置。



3、使用procmon.exe工具监测指定文件，可以看到index.php被主进程php-cgi.exe调用。



4、使用Virbox Protector工具对php-cgi.exe进行加壳，使用DSProtector对PHP资源文件进行加密。

5、重新启动服务，程序正常运行即可。

Linux

Apache服务

场景：Linux系统上使用apache和PHP服务环境，如何使用DSProtector加密呢?

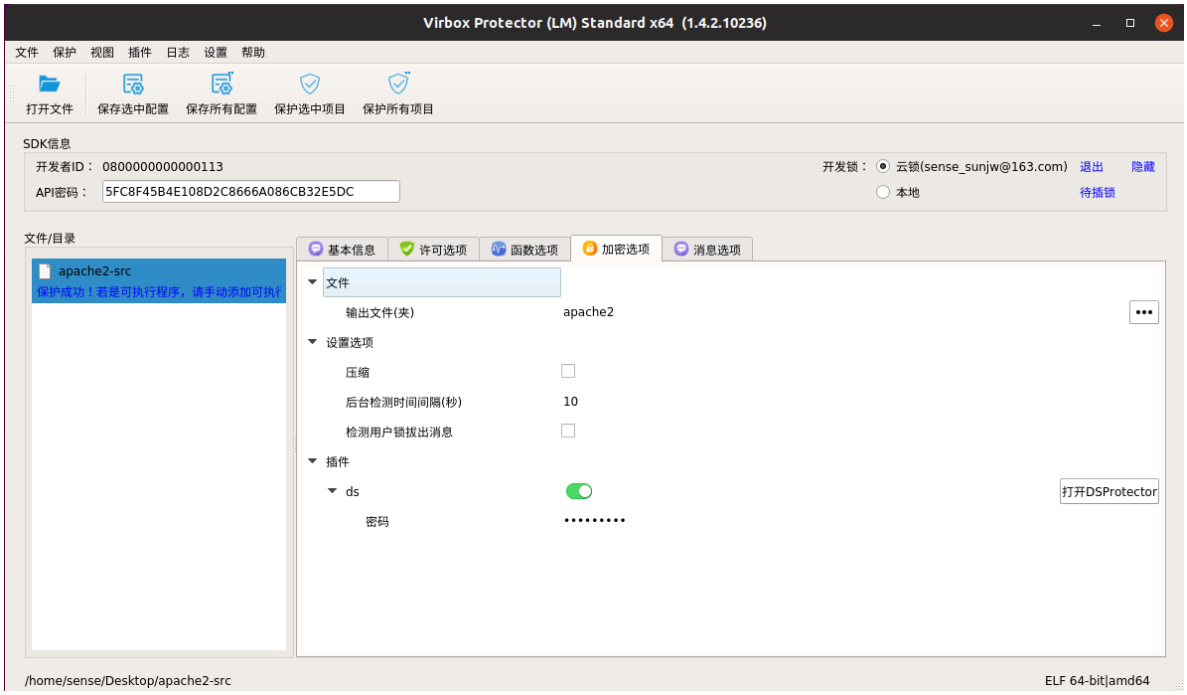
Ubuntu

1、启动Apache服务后，查看服务状态，可以看到apache2服务进程的位置及PID。

```
sense@sense:~/Desktop$ sudo systemctl status apache2.service
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
   Active: active (running) since Thu 2020-01-02 15:38:54 CST; 59min ago
  Process: 6664 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCE
 Main PID: 6686 (apache2)
    Tasks: 55 (limit: 4651)
   Memory: 7.7M
   CGroup: /system.slice/apache2.service
           ├─6686 /usr/sbin/apache2 -k start
           ├─6688 /usr/sbin/apache2 -k start
           └─6689 /usr/sbin/apache2 -k start

1月 02 15:38:54 sense systemd[1]: Starting The Apache HTTP Server...
1月 02 15:38:54 sense apachectl[6664]: AH00558: apache2: Could not reliably dete
1月 02 15:38:54 sense systemd[1]: Started The Apache HTTP Server.
lines 1-15/15 (END)
```

2、使用Virbox Protector工具对进程文件进行加壳保护。



3、使用DSProtector工具对资源文件进行加密。若DSProtector工具是以普通用户权限启动，则对root权限的文件直接加密会失败。

4、将加壳后的进程文件和加密后的资源文件放回程序原来的位置，启动Apache服务 。

5、无许可状态下，服务无法启动成功。

```
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: failed (Result: exit-code) since Thu 2020-01-02 17:41:21 CST; 30s ago
  Process: 8637 ExecStop=/usr/sbin/apachectl stop (code=exited, status=32)
  Process: 8654 ExecStart=/usr/sbin/apachectl start (code=exited, status=32)
 Main PID: 6686 (code=exited, status=0/SUCCESS)

1月 02 17:41:21 sense systemd[1]: Starting The Apache HTTP Server...
1月 02 17:41:21 sense apachectl[8654]: Unable to init server: Could not connect: Connection r
1月 02 17:41:21 sense apachectl[8654]: Warnning: Linux, show GTK UI failed, now only print me
1月 02 17:41:21 sense apachectl[8654]: 错误码13000020:找不到许可,
1月 02 17:41:21 sense apachectl[8654]: 运行此程序需要读取0号许可，但是在此计算机上找不到许可
1月 02 17:41:21 sense apachectl[8654]: Action 'start' failed.
1月 02 17:41:21 sense apachectl[8654]: The Apache error log may have more information.
1月 02 17:41:21 sense systemd[1]: apache2.service: Control process exited, code=exited status
1月 02 17:41:21 sense systemd[1]: apache2.service: Failed with result 'exit-code'.
1月 02 17:41:21 sense systemd[1]: Failed to start The Apache HTTP Server.
```

6、插入硬件锁或登录云锁，有许可状态下，服务正常启动，网站能正常运行。

CentOS

1、启动Apache服务后，查看服务状态，可以看到httpd服务进程的位置及PID。

```
huangxq@bogon sbin]$ service httpd start
Redirecting to /bin/systemctl start  httpd.service
huangxq@bogon sbin]$ service httpd status
Redirecting to /bin/systemctl status  httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor prese
t: disabled)
   Active: active (running) since Tue 2019-11-05 15:54:24 CST; 11min ago
     Docs: man:httpd(8)
           man:apachectl(8)
  Process: 16378 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=1/FAI
LURE)
 Main PID: 16538 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/s
ec"
   CGroup: /system.slice/httpd.service
           ├─16538 /usr/sbin/httpd -DFOREGROUND
           ├─16578 /usr/sbin/httpd -DFOREGROUND
           ├─16579 /usr/sbin/httpd -DFOREGROUND
           ├─16580 /usr/sbin/httpd -DFOREGROUND
           ├─16581 /usr/sbin/httpd -DFOREGROUND
           └─16582 /usr/sbin/httpd -DFOREGROUND
```

2、使用Virbox Protector工具对httpd进程加壳保护。使用DSProtector工具对资源文件进行加密。

3、若DSProtector工具是以普通用户权限启动，则对root权限的文件直接加密会失败。

php-fpm服务

场景：使用nginx服务和php-fpm服务运行PHP文件。

1、启动php-fpm服务后，查看服务状态，可以看到服务进程的位置及PID。

```
[root@localhost sense]# systemctl start php-fpm.service
[root@localhost sense]# systemctl status php-fpm.service
● php-fpm.service - The PHP FastCGI Process Manager
   Loaded: loaded (/usr/lib/systemd/system/php-fpm.service; disabled; vendor pre
set: disabled)
   Active: active (running) since Fri 2020-01-17 18:29:32 CST; 5s ago
 Main PID: 8580 (php-fpm)
   Status: "Ready to handle connections"
    Tasks: 6
   Memory: 31.7M
   CGroup: /system.slice/php-fpm.service
           ├─8580 php-fpm: master process (/etc/php-fpm.conf)
           ├─8585 php-fpm: pool www
           ├─8586 php-fpm: pool www
           ├─8587 php-fpm: pool www
           ├─8588 php-fpm: pool www
           └─8589 php-fpm: pool www

Jan 17 18:29:31 bogon systemd[1]: Starting The PHP FastCGI Process Manager...
Jan 17 18:29:32 bogon systemd[1]: Started The PHP FastCGI Process Manager.
[root@localhost sense]#
```

2、使用Virbox Protector工具对httpd进程加壳保护。使用DSProtector工具对资源文件进行加密。

3、若DSProtector工具是以普通用户权限启动，则对root权限的文件直接加密会失败。

Lua文件

平台：Ubuntu18.04

文件类型：lua

编辑器：luajit或lua

【注】luajit作为编译器，就直接对luajit最终的可执行文件进行加壳保护，不能对软连接的luajit进行加壳保护。

1、运行方式：./luajit-2.04 demo.lua

```
sense@sense:/usr/local/luajit/bin$ ./luajit-2.0.4 '/home/sense/Desktop/DS_Lua/de
mo.lua'
Line 1 - a not equal to b
Line 2 - a not equal to  b
Line 3 - a greater than or equal to b
Line 4 - a greater than b
```

2、直接Virbox Protector工具对luajit-2.04 主进程加壳，DSProtector对资源文件进行加密。

3、程序可以正常运行。

总结：针对直接调用主进程运行的程序，比如go、perl、R、Ruby等脚本语言，可以直接使用Virbox Protector工具对主进程加壳保护，再使用DSProtector对资源文件进行加密的方式。

常见问题

exe4j打包为何要更改java环境?

现象描述

使用exe4j将jar包打包成exe后，运行exe程序，将exe产生的临时文件拷贝到其他目录后打开，文件是未加密状态。

问题分析

由于exe4j打包时依赖java环境，若打包过程中jvm.dll和加密后的jar包密钥匹配，则会在打包过程中将jar包解密，打包成功的exe运行产生的临时文件也将是被解密过后的，即原文件，这样无法起到保护作用。

解决方案

参考描述exe4j打包流程的方案二。

Linux Apache/Nginx 加壳后程序启动失败

【注意】以下只列举了Apache和Nginx服务修改情况，针对php-fpm或tomcat等服务修改方式使用方式一样。

现象描述

开发者开发的网站类的组件或应用依赖于 Apache/Nginx 服务，在启动 Apache/Nginx 等网站服务时，以子进程的方式启动开发者的组件功能。开发者使用**Virbox Protector LM体系**工具，加壳后的程序无法通过Apache/Nginx 服务启动。但单独运行进程时正常，可以正常找到许可。

问题分析

Virbox Protector LM体系工具保护后的程序中所内置的API（Runtime API）需要通过 /tmp 目录下的文件句柄与Senseshield 服务通信，检查授权。

Apache/Nginx 等网站服务，考虑到安全性，默认配置使用私有的的 /tmp 目录，与系统的 /tmp 目录隔离，避免黑客通过 /tmp 目录的漏洞攻击网站服务器。以为 Apache/Nginx 服务所使用的 /tmp 目录与 Senseshield 进程所使用 /tmp 目录不同，导致无法实现进程间通信，且通过开发者应用的日志可以发现，Apache/Nginx 启动时API返回的错误为 0x02000003 （连接失败）。

解决方案

Apache

经查到在linux上apache2.service配置上，privateTmp=true说明这个服务使用到tmp目录是默认会创建一个私有文件夹，加壳后的文件运行时使用到SS也会默认创建一个私有文件夹，导致加壳后的文件不通过SS，所以会一直提示连接失败，将privateTmp=flase就不会出现此现象。

- 从返回的信息可以看到 apache2的服务配置文件路径：/lib/systemd/system/apache2.service

```
sense@sense:~/Desktop$ sudo systemctl status apache2.service
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset:
   Active: active (running) since Thu 2020-01-02 15:38:54 CST; 59min ago
  Process: 6664 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCE
 Main PID: 6686 (apache2)
    Tasks: 55 (limit: 4651)
   Memory: 7.7M
   CGroup: /system.slice/apache2.service
           ├─6686 /usr/sbin/apache2 -k start
           ├─6688 /usr/sbin/apache2 -k start
           └─6689 /usr/sbin/apache2 -k start

1月 02 15:38:54 sense systemd[1]: Starting The Apache HTTP Server...
1月 02 15:38:54 sense apachectl[6664]: AH00558: apache2: Could not reliably dete
1月 02 15:38:54 sense systemd[1]: Started The Apache HTTP Server.
```

- 修改配置 PrivateTmp=true 为 PrivateTmp=false，保存配置文件。

```
sense@sense:~/Desktop$ cat /lib/systemd/system/apache2.service
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
Environment=APACHE_STARTED_BY_SYSTEMD=true
ExecStart=/usr/sbin/apachectl start
ExecStop=/usr/sbin/apachectl stop
ExecReload=/usr/sbin/apachectl graceful
PrivateTmp=false    True改为false
Restart=on-abort

[Install]
WantedBy=multi-user.target
```

- 重新加载服务配置
  - 启动服务 systemctl start apache2.service，提示需要执行 daemon-reload 命令重新加载。
  - 执行 systemctl daemon-reload 。
  - 再次启动服务，服务则正常运行。

Nginx

- 查找 Nginx 服务配置文件路径。

```
[huangxq@bogon sbin]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor prese
t: disabled)
   Active: active (running) since Thu 2020-01-02 17:03:14 CST; 5s ago
  Process: 20210 ExecStart=/usr/sbin/nginx (code=exited, status=0/SUCCESS)
  Process: 20208 ExecStartPre=/usr/sbin/nginx -t (code=exited, status=0/SUCCESS)
  Process: 20202 ExecStartPre=/usr/bin/rm -f /run/nginx.pid (code=exited, status
=0/SUCCESS)
 Main PID: 20213 (nginx)
   CGroup: /system.slice/nginx.service
           ├─20213 nginx: master process /usr/sbin/nginx
           └─20214 nginx: worker process

Jan 02 17:03:13 bogon systemd[1]: Starting The nginx HTTP and reverse proxy.....
Jan 02 17:03:14 bogon nginx[20208]: nginx: the configuration file /etc/ngin...ok
Jan 02 17:03:14 bogon nginx[20208]: nginx: configuration file /etc/nginx/ng...ul
Jan 02 17:03:14 bogon systemd[1]: Started The nginx HTTP and reverse proxy ...r.
Hint: Some lines were ellipsized, use -l to show in full.
```

从返回的信息可以看到 Nginx 的服务配置文件路径：/etc/systemd/system/nginx.service

- 修改配置 PrivateTmp=true 为 PrivateTmp=false，保存配置文件。

```
[huangxq@bogon sbin]$ sudo cat /usr/lib/systemd/system/nginx.service
[Unit]
Description=The nginx HTTP and reverse proxy server
After=network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
# Nginx will fail to start if /run/nginx.pid already exists but has the wrong
# SELinux context. This might happen when running `nginx -t` from the cmdline.
# https://bugzilla.redhat.com/show_bug.cgi?id=1268621
ExecStartPre=/usr/bin/rm -f /run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecReload=/bin/kill -s HUP $MAINPID
KillSignal=SIGQUIT
TimeoutStopSec=5
KillMode=process
PrivateTmp=true   true改为flase

[Install]
WantedBy=multi-user.target
```

- 重新加载服务配置
    - 启动服务 systemctl start nginx ，提示需要执行 daemon-reload 命令重新加载。
    - 执行 systemctl daemon-reload
    - 再次启动服务，服务则正常运行。

如何设置服务依赖项顺序

场景描述：若针对服务开机自启的情况时，要保证服务启动在ss服务启动之后。以php-fpm的服务设置为例，如图所示：

解决方式：在service文件中After参数后加senseshield.service，然后保存后重新启动服务即可。

```
File  Edit  View  Search  Terminal  Help
[Unit]
Description=The PHP FastCGI Process Manager
After=syslog.target network.target senseshield.service

[Service]
Type=notify
PIDFile=/var/run/php-fpm/php-fpm.pid
EnvironmentFile=/etc/sysconfig/php-fpm
ExecStart=/usr/sbin/php-fpm --nodaemonize --fpm-config /etc/php-fpm.conf
ExecReload=/bin/kill -USR2 $MAINPID
PrivateTmp=flase

[Install]
WantedBy=multi-user.target
```

uncompyle6可以反编译出加密后pyc的源码

现象描述

用户使用uncompyle6对pyc文件进行反编译得到的文件是源码。

问题分析

1、uncompyle6是原生python的自带的反编译器，uncompyle6运行需要调用主进程python.exe。

2、加壳后的python.exe和加密后的pyc是绑定的，比如python.exe用Virbox Protector LM工具保护后，没有许可的情况下uncompyle6将无法运行的，当有许可时，相当于给python.exe授权可以解密pyc，uncompyle6功能可以使用，则能反编译出源码。

解决方案

建议直接对py文件进行加密或者将pyc文件转为其他形式文件后再进行保护。

java多个环境

现象描述

使用Apache Ambari，jar包以服务的形式运行，直接以启动脚本的方式运行服务，对java加壳，对jar包加密后，网页无法运行。

问题分析

1、首先查看脚本中服务如何启动的，比如服务脚本语言为python，python中调用的java，查看指定java的位置。

2、将正在运行的java进程全部kill。

3、再次启动服务后，查看java进程，查看java的路径和加壳后java的路径是否一致。

4、系统上有多个java，主进程保护错误，导致网页无法正常运行。

解决方案

对服务中调用的java主进程加壳，若DS密码不变，则jar包不需要重新进行加密。