Virbox Protector使用说明

产品介绍

软件自动保护工具 Virbox Protector，简称独立加壳工具，是深思数盾科技股份有限公司经过多年技术深耕开发的一款高强度自动保护（加密）工具。Virbox Protector 集自动代码移植、混淆、代码加密等于一身，无需编程就能达到极高的保护强度，是业界领先的软件保护工具。

版本对比

产品名称	描述	收费模式	目标平台	备注
Virbox Protector	独立版软件加壳工具，无 Virbox LM 许可产品关联。	收费，各平台需要单独购买许可（有许可限制）	Windows、Linux、macOS、ARM Linux 和Android	将自动安装 Virbox 客户端和反黑引擎用于自我保护
Virbox Protector Trial	在正式版独立壳的功能基础上，加壳后的程序会有 7天的试用期限。	免费（有许可限制）	Windows、Linux、macOS、ARM Linux 和Android	将自动安装 Virbox 客户端和反黑引擎用于自我保护
Virbox Protector (LM) Standard	Virbox LM SDK 中自带，必须使用精锐5/云/软许可，支持 macOS、Linux、Windows 系统。(不支持 ARM-Linux、Android 设备及其它特殊平台等)	免费	Windows、Linux、macOS	不带反黑引擎和 Virbox 客户端
Virbox Protector (LM) Professional	在标准版基础上，增加 ARM-Linux，Android SO 和 Android Unity3D的保护。	标准版之外每个平台单独购买许可	Windows、Linux、macOS、ARM Linux 和Android	将自动安装 Virbox 客户端和反黑引擎用于自我保护
Virbox Protector (Moway) Standard	魔锐加壳工具，只能和魔锐绑定使用，Virbox Moway SDK 中自带。	免费	Windows、Linux	不带反黑引擎和 Virbox 客户端
Virbox Protector (Moway) Professional	在标准版基础上，支持混淆、虚拟化和代码加密高级功能。	收费（有许可限制）	Windows、Linux	将自动安装 Virbox 客户端和反黑引擎用于自我保护

支持列表

支持多种架构，并且支持多种语言编译的程序，如图所示：

文件类型	支持系统	架构	语言
.NET	Windows	x86、x64	VB、C#等
.NET Core3	Windows、Linux、macOS	x86、x64	C#、VB.net
PE	Windows	x86、x64	C/C++、Delphi、PB、BCB等
Unity3D	Windows、Linux、macOS、Android	x86、x64、ARM32	C#等
ELF	Linux、Android	x86、x64、ARM32、ARM64	C/C++等
Mach-O	macOS	x64	C/C++、Objective-C、Swift
java	windows	x86、x64	java

获取更多帮助

您可以通过如下方式获取更多的帮助信息

官方网址：https://lm.virbox.com/

客服电话：010-56730936

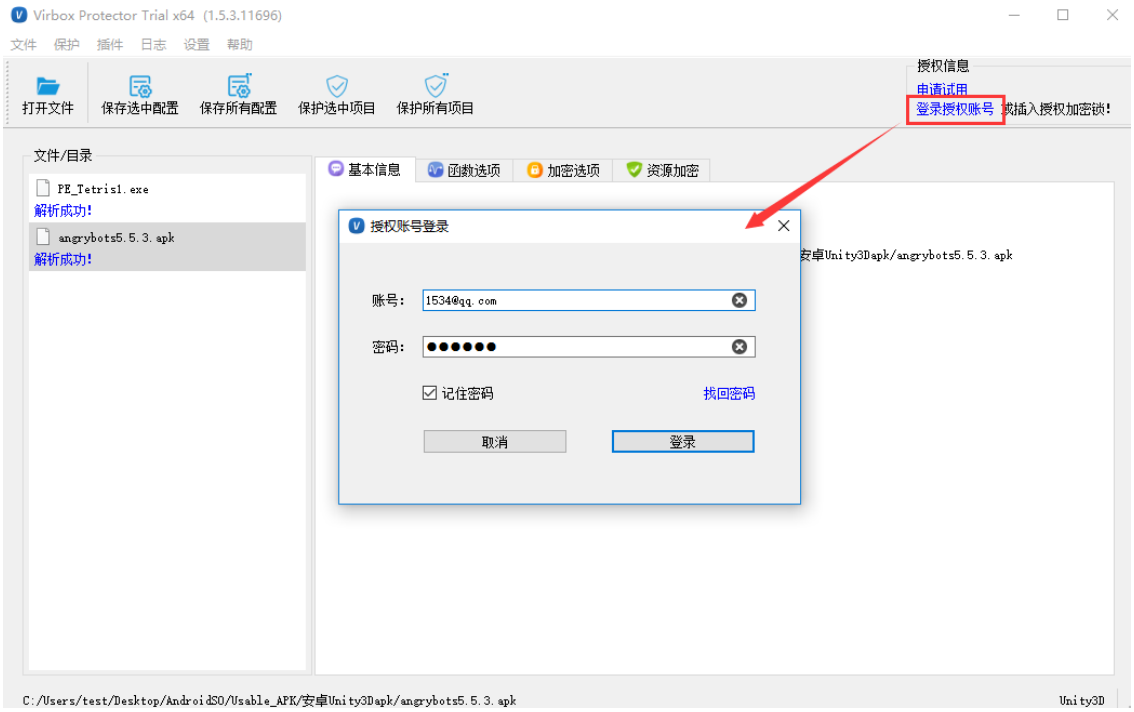
公司地址：北京市海淀区西北旺东路 10 号院 5 号楼软件园二期互联网创新中心 C 区 510

快速入门

Virbox Protector Trial

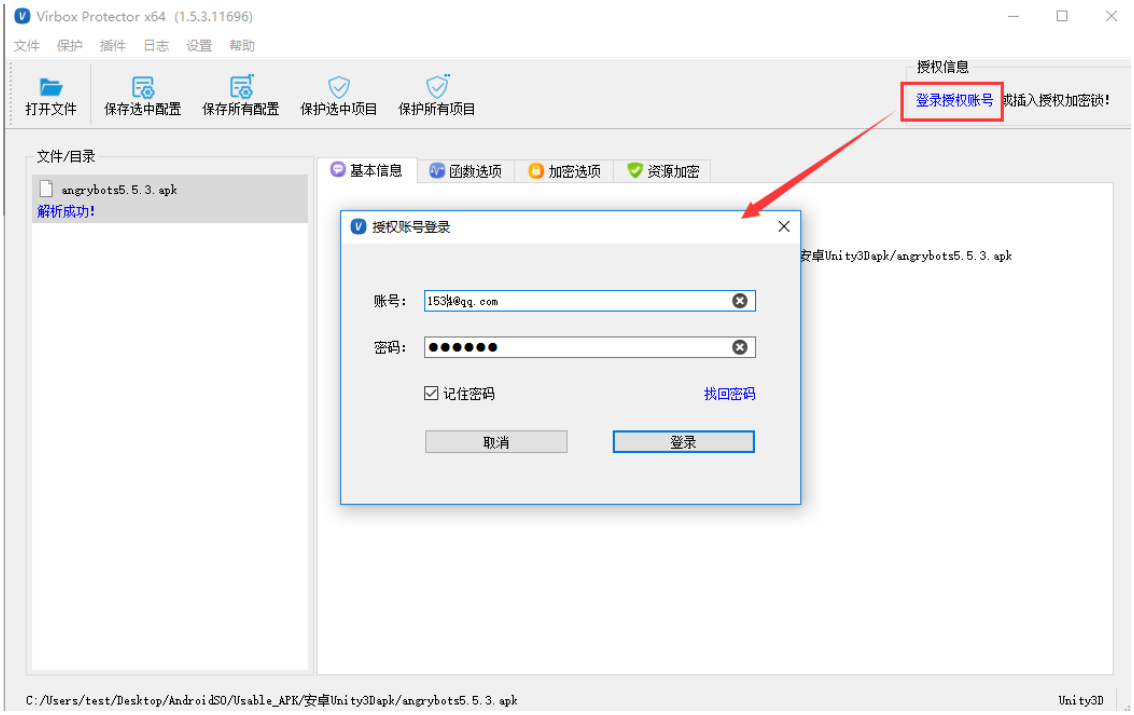
1. 在 Virbox 网站 上获取。
2. 获取到Virbox Protector Trial版本，安装成功后打开进入到Virbox Protector Trial主界面。
3. 若没有授权账号，点击申请试用按钮，进入申请试用网址页面，填写试用表后提交申请，即可获取授权账号。

4. 若已经申请过拿到授权，点击“登录授权账号”按钮，登录账号后即可对文件进行使用。



Virbox Protector

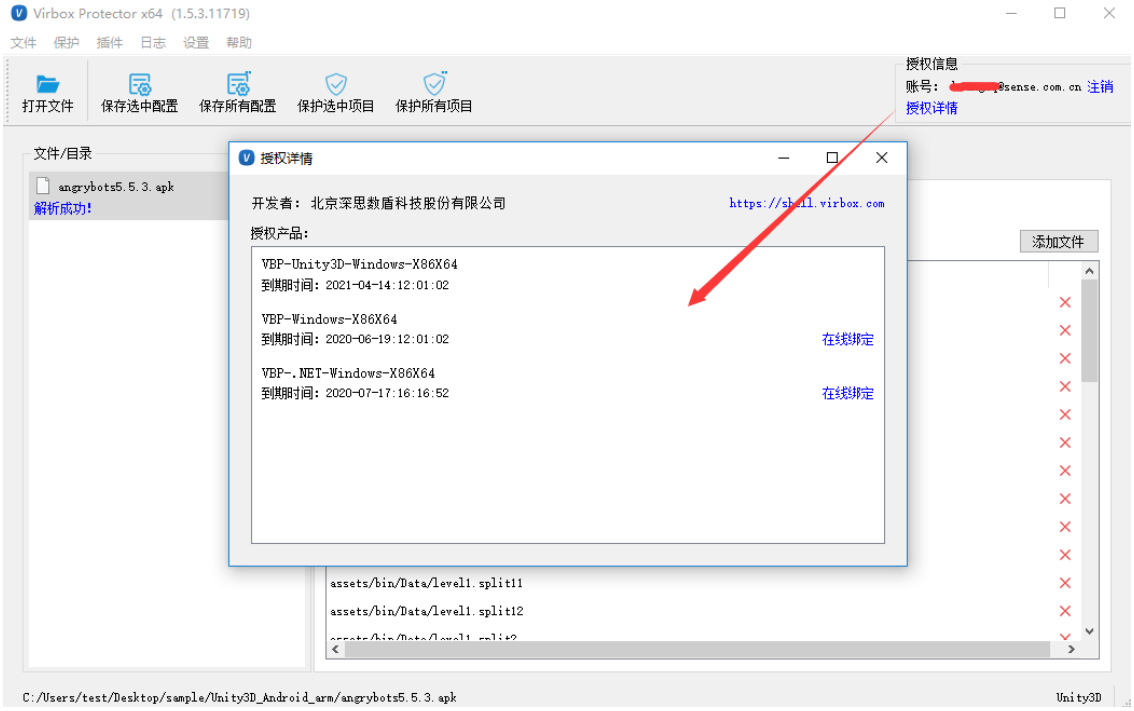
1. 需要联系**深思数盾销售人员**获取版本，授权账号为“申请试用”时的账号。
2. Virbox Protector 版本为正式版，点击“登录授权账号”按钮进行登录账号。



3. 授权账号登陆成功。

   1）授权详情，可以查看到该账号当前的授权信息。

2）注销，账号注销后，授权详情里的所有绑定信息均会解绑。



4. 若有许可的状态下即可对程序进行加壳保护。

Virbox Protector 界面使用

安装目录结构

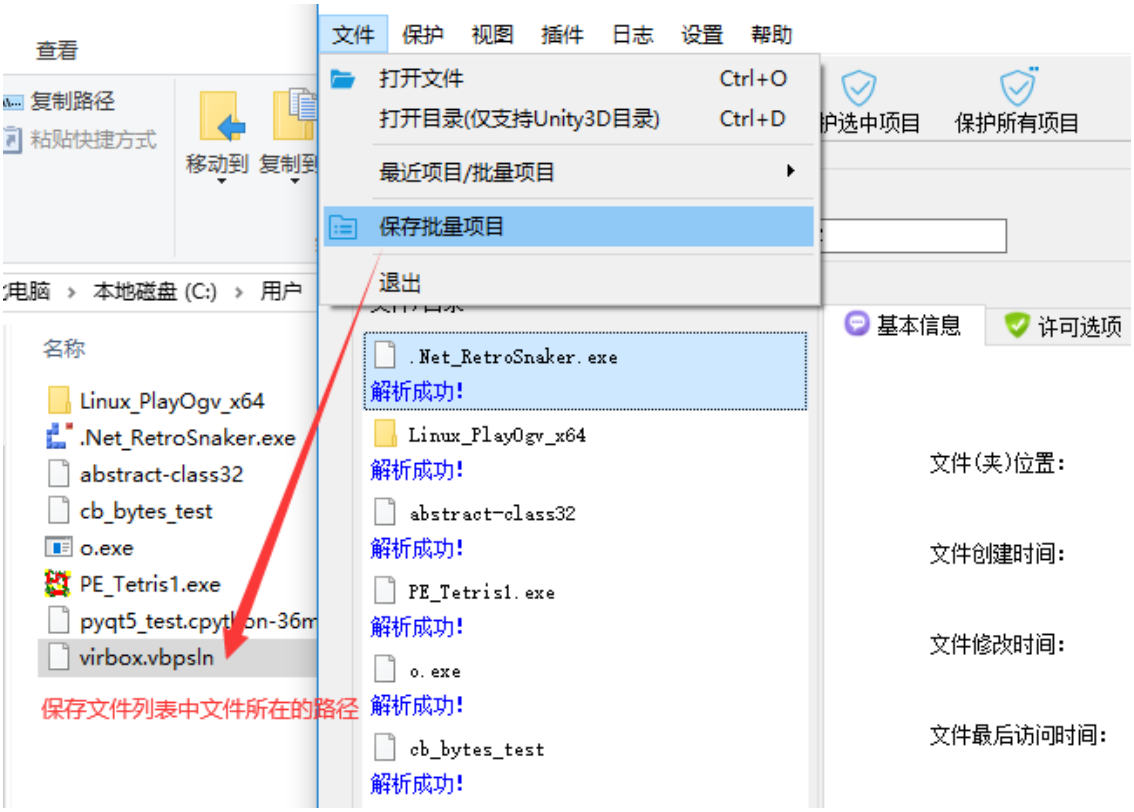
如下所示：

```
├─bin
│  ├─virboxprotector.exe
│  ├─virboxprotector_con.exe
│  └─dsprotector_con.exe
├─example
│  ├─plugin
│  │  └─demo
│  │      └─src
│  └─sdk
├─help
├─plugin
│  ├─anti
│  └─ds
└─sdk
```

使用流程

1. 授权登陆成功后，将文件拖入到加壳工具，进行信息配置，保护文件。

2. 可对文件列表中的文件进行批量保存。



3. 文件加密保护成功，可运行或发布保护过后的文件。

Virbox Protector 界面功能

工具栏

工具栏处显示打开文件、保存选中配置等快捷按钮，在工具栏处右键点击还可以隐藏工具栏。

1. 保存选中配置：指保存选中文件的设置的函数选项、加密选项等配置信息。
2. 保存所有配置：指保存文件列表中所有文件的设置的函数选项、加密选项等配置信息。
3. 保护选中项目：指将选中文件进行加密保护。
4. 保护所有项目：指将文件列表中所有文件进行加密保护。

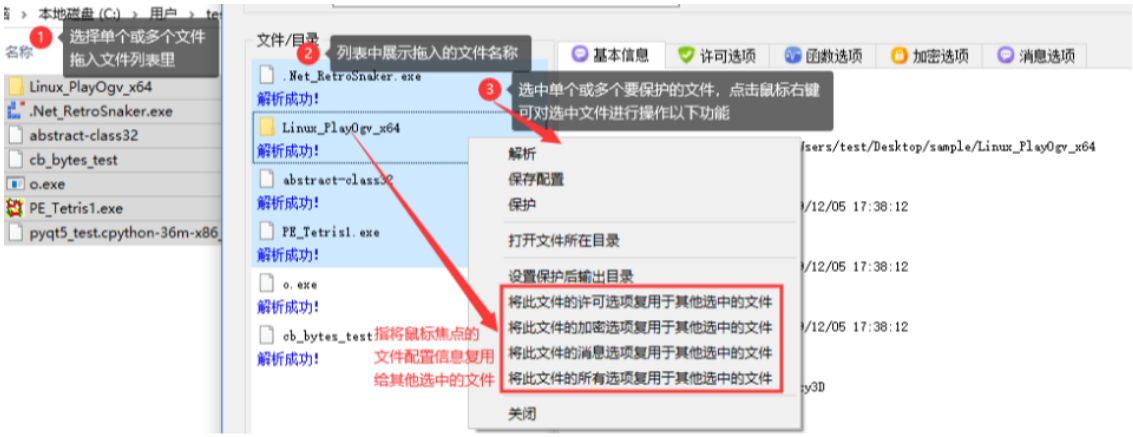
文件目录列表

展示被保护文件的信息列表。

1. 将文件拖入到 Virbox Protector 工具界面，工具界面展示文件信息。
2. 鼠标焦点的文件右侧会显示程序的基本信息、函数选项、加密选项等信息，手动进行填写所需信息。
3. 选中单个或多个文件，点击鼠标右键，在弹框列表中可对选中的文件进行批量解析、保存配置和保护等功能。

   1）打开文件所在目录：指打开鼠标焦点的文件所在目录。

   2）设置保护后输出目录：指将选中文件保护后输出的路径更改到指定的目录。



基本信息

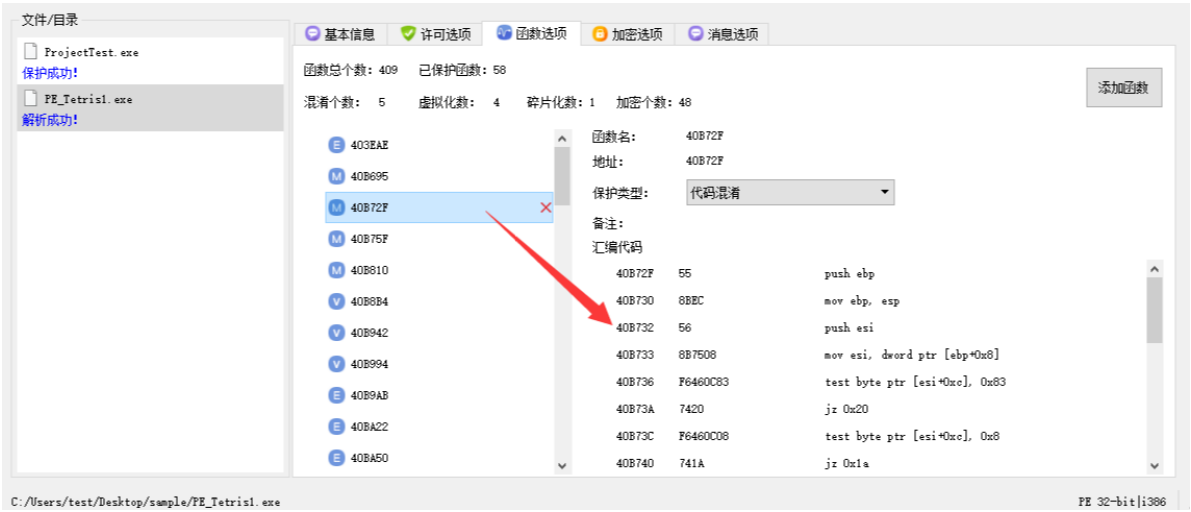
主要显示文件(夹)位置、文件创建时间、文件修改时间、文件最后访问时间和文件类型信息。

函数选项

需要保护的具有重要价值的函数块，用户能够选择混淆、虚拟化、碎片化和代码加密的保护方式。

查看函数的详细信息

鼠标点击函数保护列表中的函数，在右侧的工作区窗口中展示函数的详细信息，包括函数的保护方式、函数名、函数的地址和汇编代码的展示。

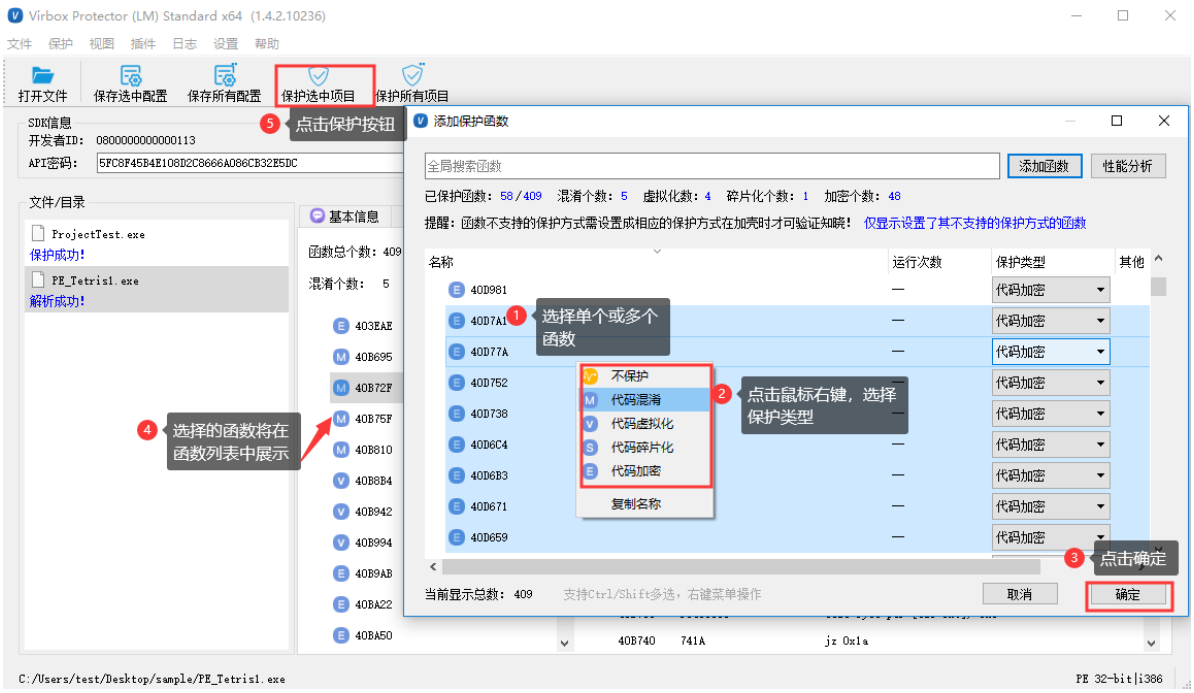


添加函数

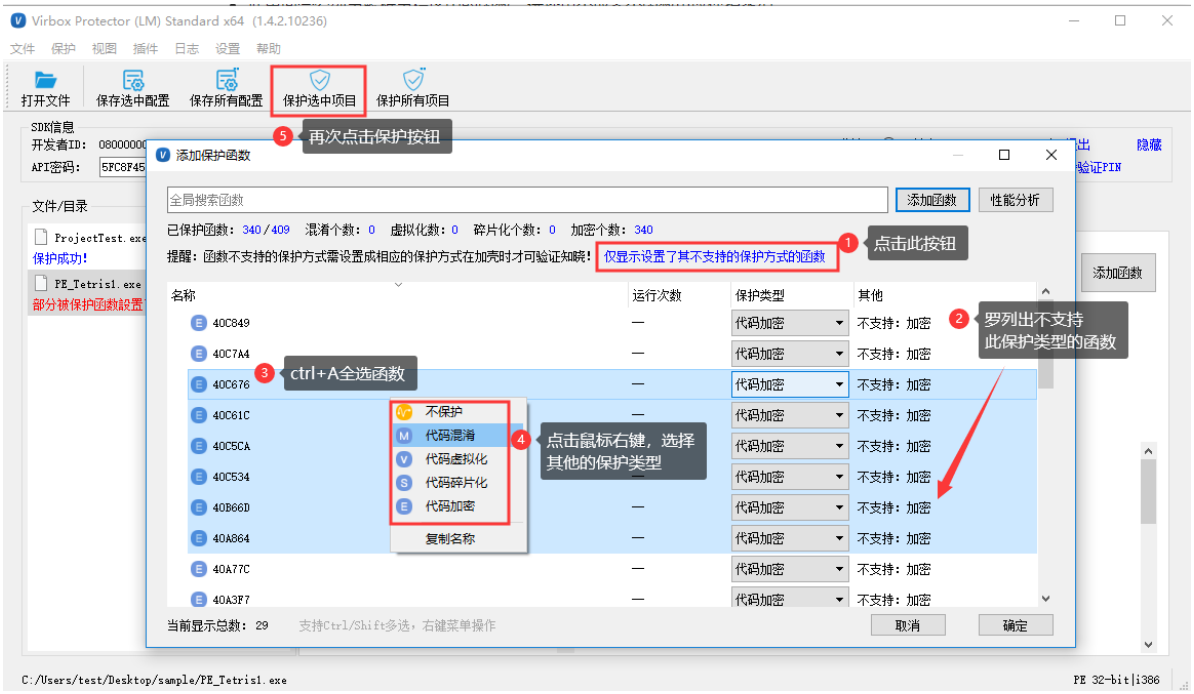
1. 点击函数选项右侧【添加函数】按钮，进入到添加保护函数界面。
2. 此界面会罗列出解析出程序中的函数模块(托管代码程序和非托管代码程序有细微的差别)，选择单个或多个函数更改保护类型。

- 托管代码程序：函数名称为“命名空间+类名称+函数名称”。
- 非托管代码程序：函数名称为函数的va的值。

3. 点击确定按钮，选择的函数将在函数列表中展示，点击保护选中项目进行加密程序。如图所示：



4. 若保护过程中失败，提示“部分被保护的函数设置了不支持的保护方式“，需要将其更改为其他的保护方式后再进行加壳。如图所示：



全局搜索函数

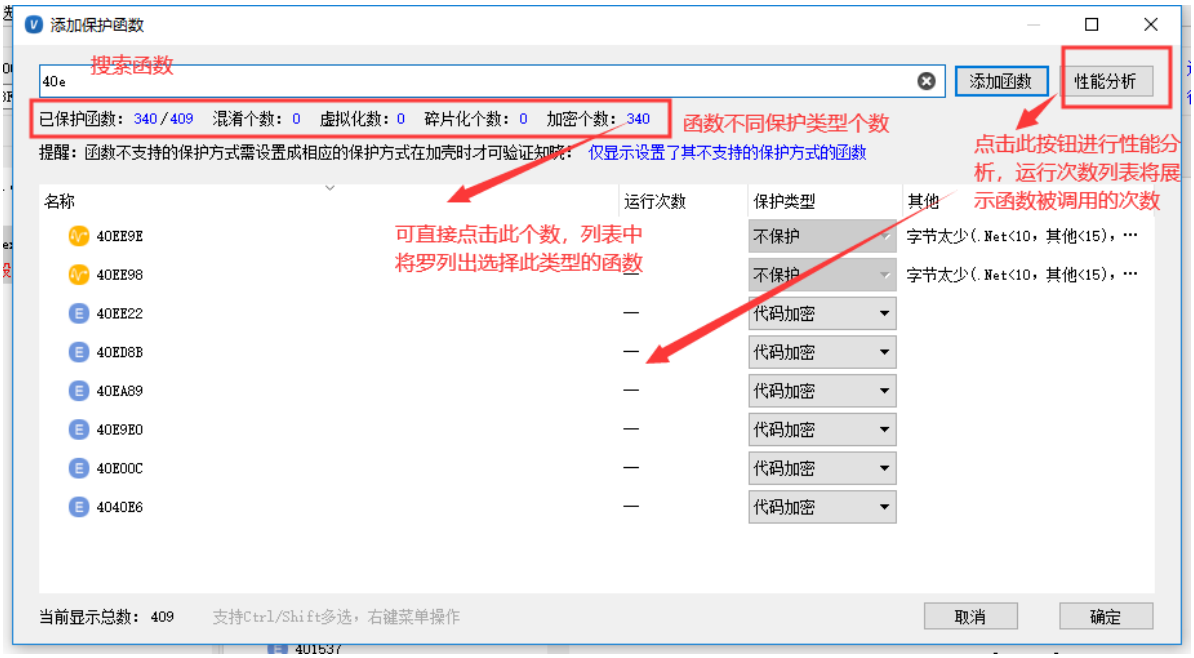
在搜索框中输入关键字，程序会列出所有名称中包含关键字的函数块，支持模糊查询。

性能分析

点击性能分析按钮，运行需要保护的程序，执行正常的业务操作，然后关闭程序后，程序中各个函数模块调用的次数即可显示在列表中，参见上图。如果当前分析的程序位为II程序，需要选择启动模块。

信息展示

可保护函数列表信息展示，显示了函数的总个数、已添加函数个数、混淆函数个数、碎片代码函数个数。



加密选项

输出文件(夹)

输出文件，可以修改程序保护后生成文件的路径和名称。

设置选项

包括导入表保护、压缩、资源保护、名称混淆等功能，主要是对文件的整体保护。

状态栏

Virbox Protector 工具的状态栏从左到右分别显示了被保护程序的文件的全路径、程序的类型以及程序的硬件机器版本。

本地程序保护

本地可执行程序包括 PE、ELF、Mach-O 文件格式。

基础保护

导入表保护

隐藏原程序中的导入表，保护程序的函数外部调用，可以达到干扰逆向分析、防脱壳的作用。

支持范围

目前仅支持 PE 格式的程序。

原理

去除原程序的导入表，将导入地址表(IAT) 替换为修复函数，由壳代码接管导入函数的跳转。

资源加密

资源加密是针对 PE 格式程序的资源进行加密的保护功能，可以防止程序中的资源信息被提取，篡改。

原理

在加壳时将 PE 格式程序中的资源抽取并加密，仅保护一些外部需要的资源（如图标、版本信息等），在程序执行时，在壳代码中再解密。

附加数据扩展

什么是附加数据？

附加数据一般是由某些编译器或打包工具，将一些数据（如音视频、数据库等）与原始的可执行程序拼接，这些数据一般在运行时被原始的程序读取，附加数据在执行时并不会直接被映射到内存中。

功能

由于加壳会改变原始程序文件，如果将附加数据直接拼接到保护后的程序，可能会导致运行时异常。

附加数据扩展使用了 Hook 手段使程序能正常读取到附加数据，另外对附加数据做了加密处理，防止数据被轻易窃取。

压缩

Virbox Protector 的压缩功能，其核心目的不是“压缩”，并非专为缩小程序体积而设计的。它真正的作用是将代码与数据段做了加密，并将原先的导入表与重定位信息隐藏了起来，再”顺便“将原先的数据做了压缩。

原理

将原始的代码段与数据包打包并压缩，将原始程序入口（OEP）替换为壳代码，运行时由壳代码将代码段与数据段还原，并进行一些重定位等操作，使程序能正常运行。

功能

防止静态反编译，防止程序被打补丁。

- **优点**

  1、能起到一层整体保护效果，可以隐藏程序的代码、数据和文件结构信息。

  2、运行效率高，仅在程序被加载时轻微的性能损失。

- **缺点**

  1、壳代码执行完毕后，代码段与数据段会还原，可以被 Dump。

内存校验

1. 内存校验是 Virbox Protector 实现的一种检测程序自身完整性的技术，可以对抗文件补丁、内存补丁、软件断点等，内存校验表和校验逻辑本身经过了自保护，以保证其安全性。
2. 内存校验运行时在程序入口，壳代码会对每个要校验的内存块进行校验，以验证其完整性，如果校验失败，则会清场退出。
3. 如果使用了 SDK 标签，则每次调用标签 VBProtectVerifyImage 时都会进行校验。

使用方法

将PE或ELF程序拖入到Virbox Protector工具中，加密选项就会显示“内存校验”选项，勾选上此选项，对程序进行保护，则保护后的程序将拥有此功能的效果。

反调试

通过平台相关API、数据结构和寄存器，检测调试器，主要防止保护后的程序被反编译工具(如gdb，windbg等)进行动态调试。

函数级保护

代码混淆

原理

代码混淆亦称花指令，是将计算机程序的代码，转换成一种功能上等价，但是难于阅读和理解的形式。

Virbox Protector 支持对 x86/arm/.net il 系列指令进行混淆。

功能

扰乱原始指令，防止静态分析。

- **优点**

  1、防反编译，代码分析难度大。

- **缺点**

  1、运行效率有损失。

反 Run Trace

> 注：反Run Trace功能主要是针对ARM架构的程序

1. Run Trace 是调试器提供的功能，用于单步执行每一条指令，并记录每条指令执行时的寄存器状态，是调试追踪和反混淆常用的手段。
2. ARM架构的程序使用了Virbox Protector代码混淆功能后，函数的指令会添加一些暗桩用于检测单步断点，检测到非法调试会执行一些错误的指令，干扰调试分析，使得程序无法正常运行或者调试器崩溃，从而进一步增强混淆后代码的安全性。

代码虚拟化

> 注：支持X86、X64和ARM架构的程序。

原理

将原始指令转换为自定义的虚拟机指令，交由配套虚拟机系统模拟执行。

功能

隐藏原始指令，防止代码逻辑分析。

- 优点

  1、保护强度高，几乎不能被分析出原始的代码逻辑。

- 缺点

  1、运行效率低。

代码加密(Native)

原理

代码加密是使用 SMC（Self-Modifying Code）技术，将原始的函数加密，在**函数**被执行时才将函数解密并执行的保护方式。

功能

防脱壳，防止直接 Dump。

- 优点

  1、运行效率高，几乎没有性能损失。

- 缺点

  1、函数执行后会解密，解密之后容易被分析。

自动化保护

目前支持pdb和map文件，若两种文件同时存在，默认识别pdb文件。

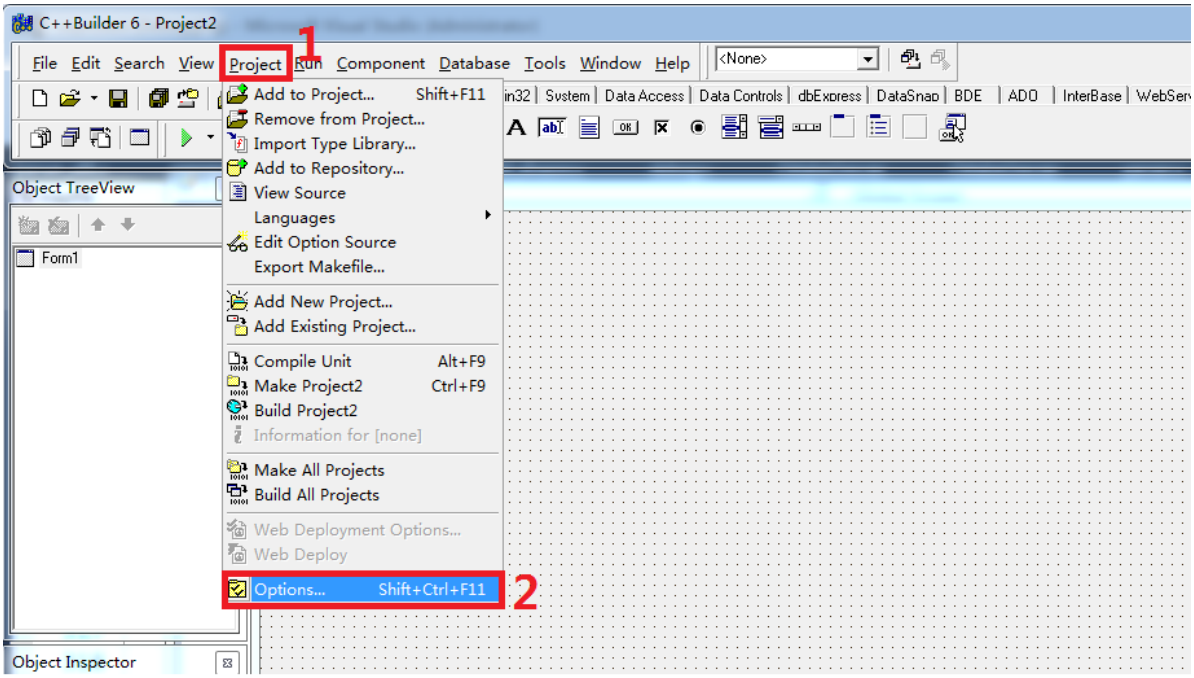
使用 map 文件

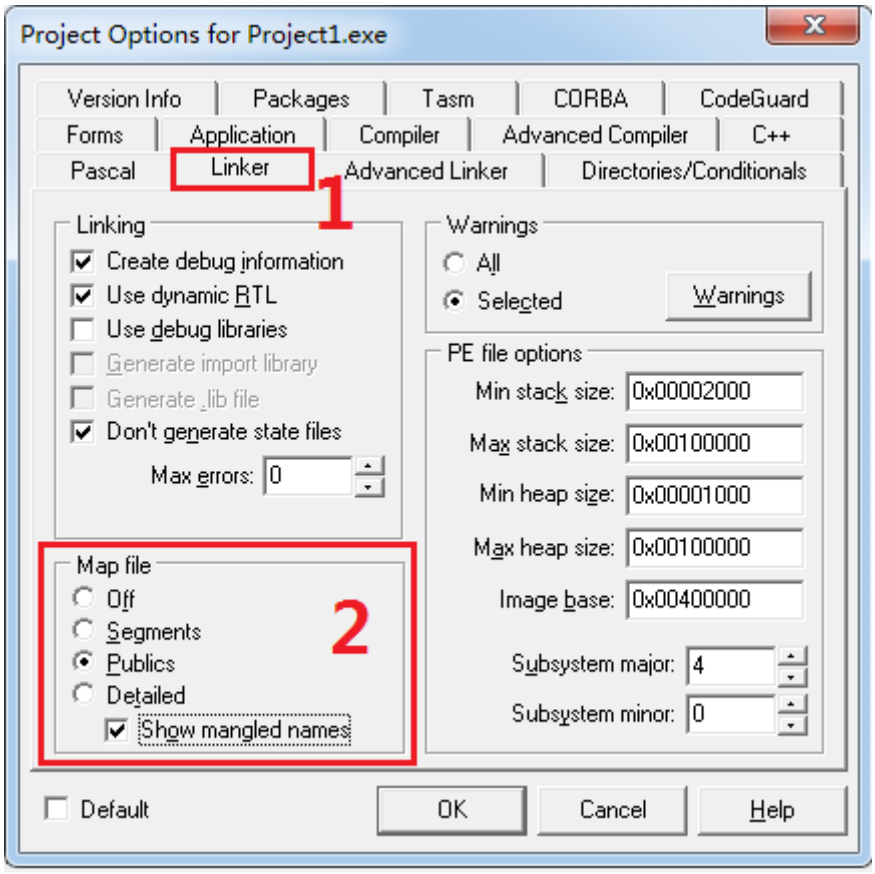
加壳工具解析PE程序时函数是以地址的方式显示，若有map文件则加壳工具解析PE程序时函数是以函数名的方式进行显示。

使用 BCB 生成 map 文件

BCB全称（Borland C++ Builder），使用C++ Builder工具生成map文件。

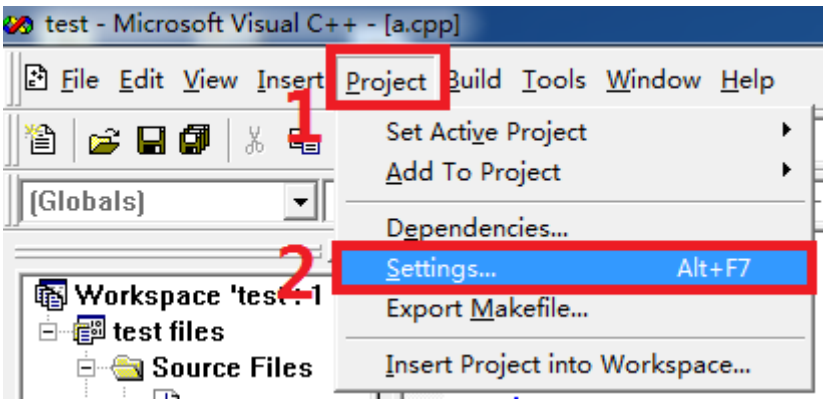
- 工程设置如下图：

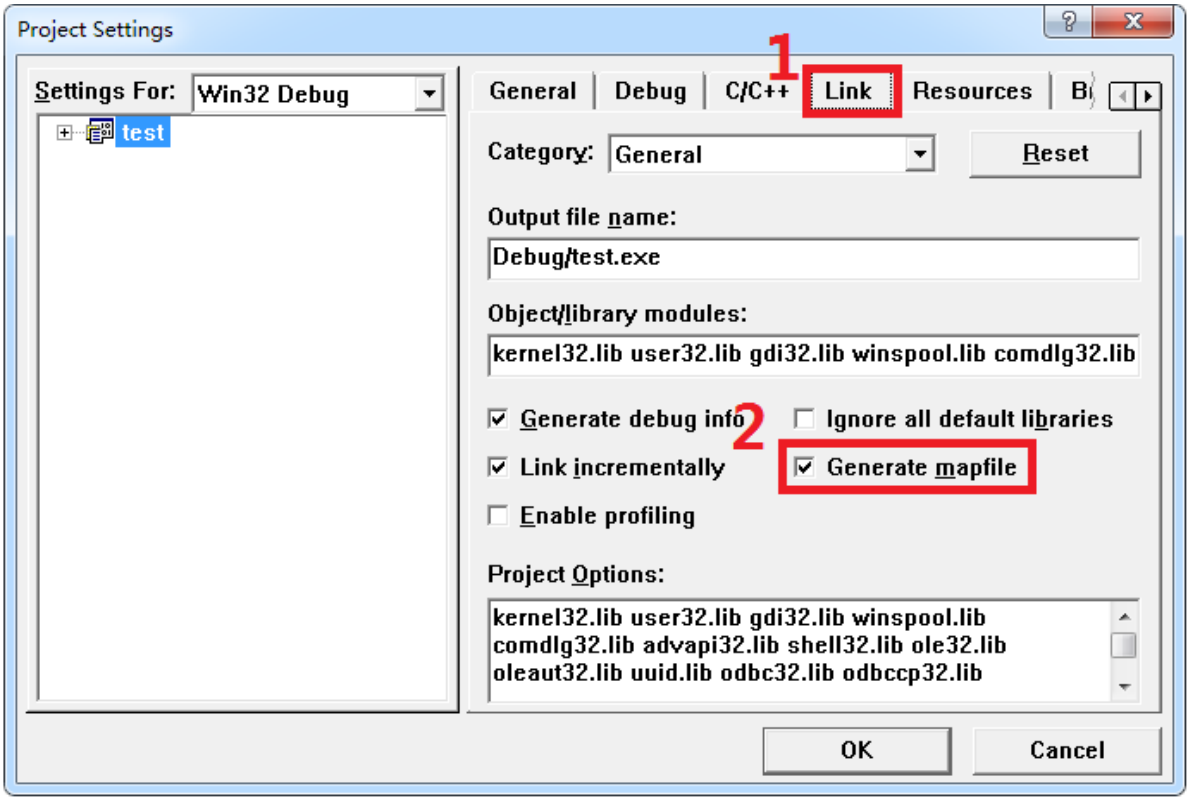




使用VC生成map文件

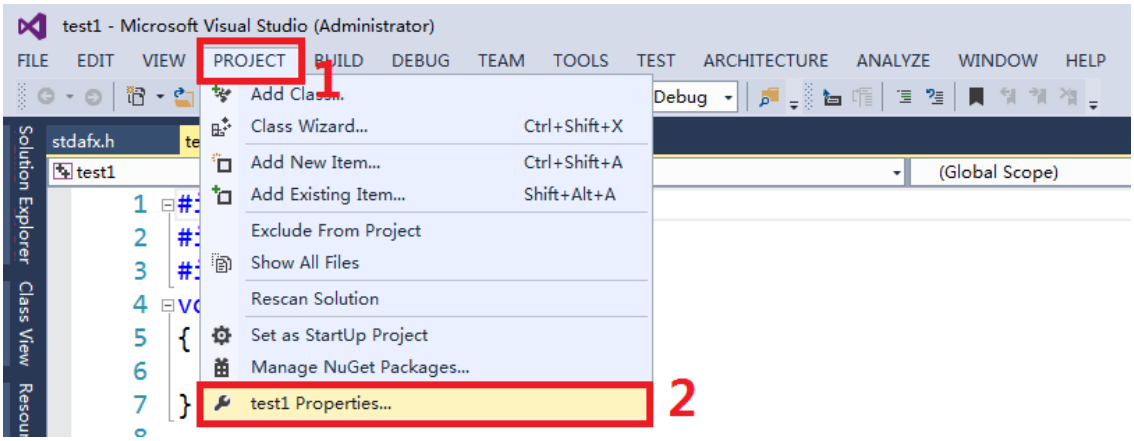
- 工程设置如下图：

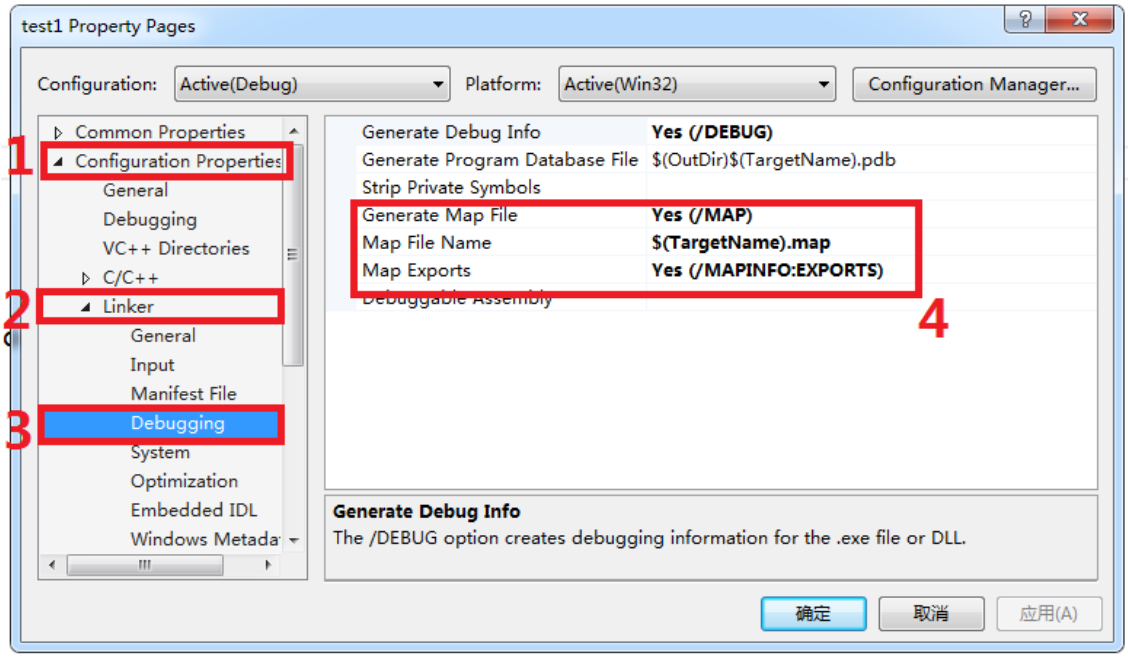




使用 VS 生成 map 文件

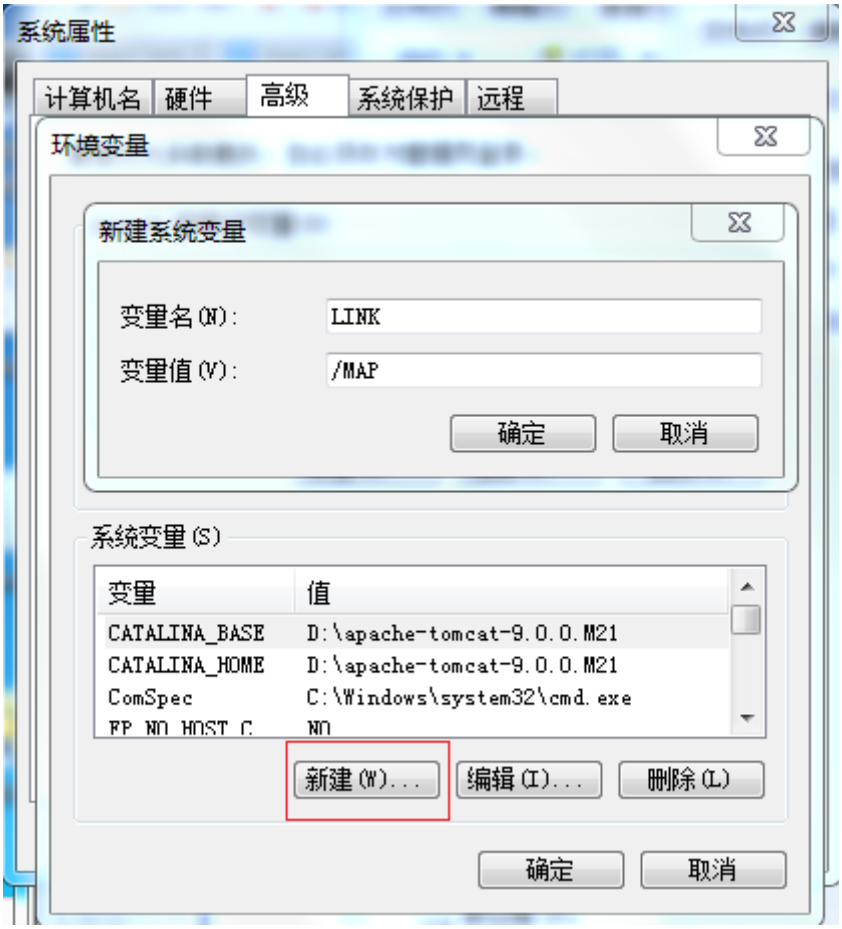
- 工程设置如下图：





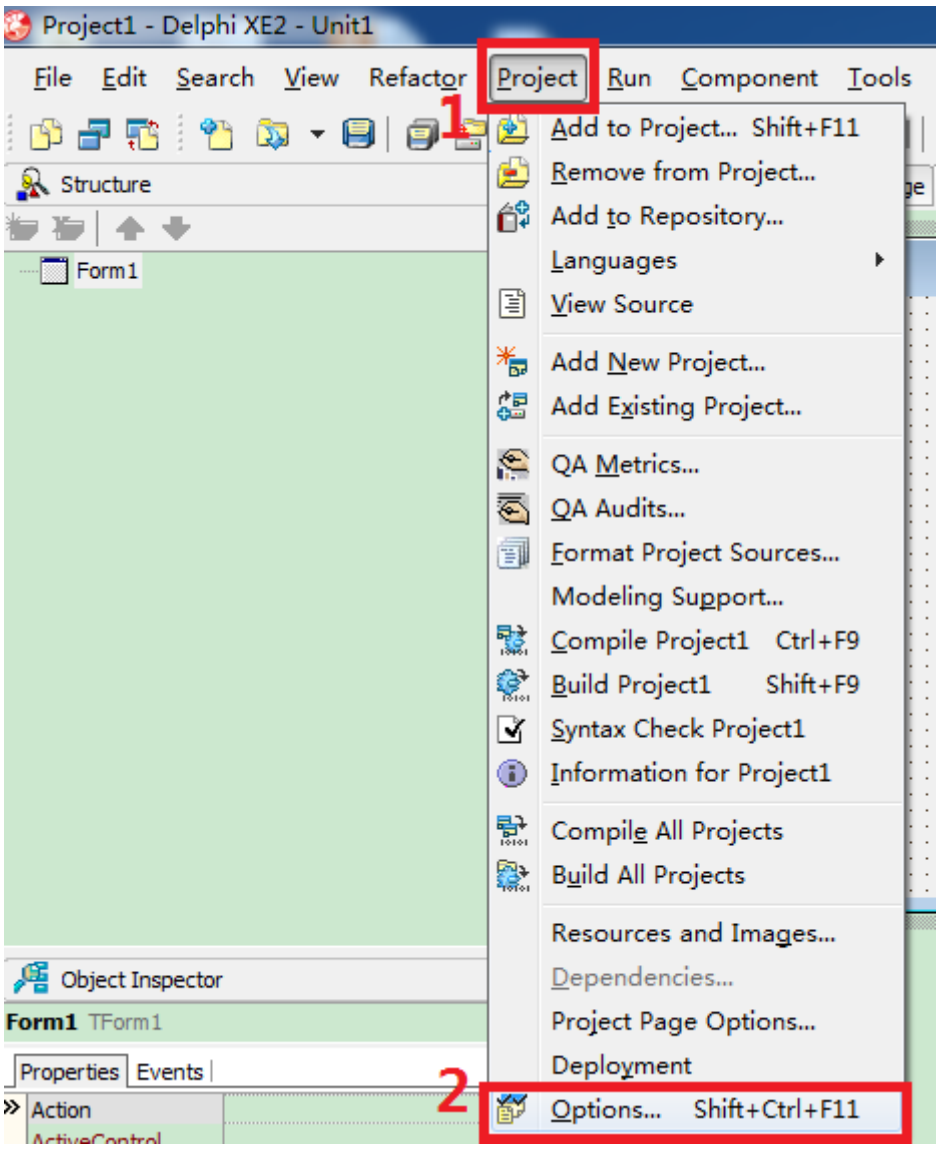
使用 VB6.0 生成 map 文件

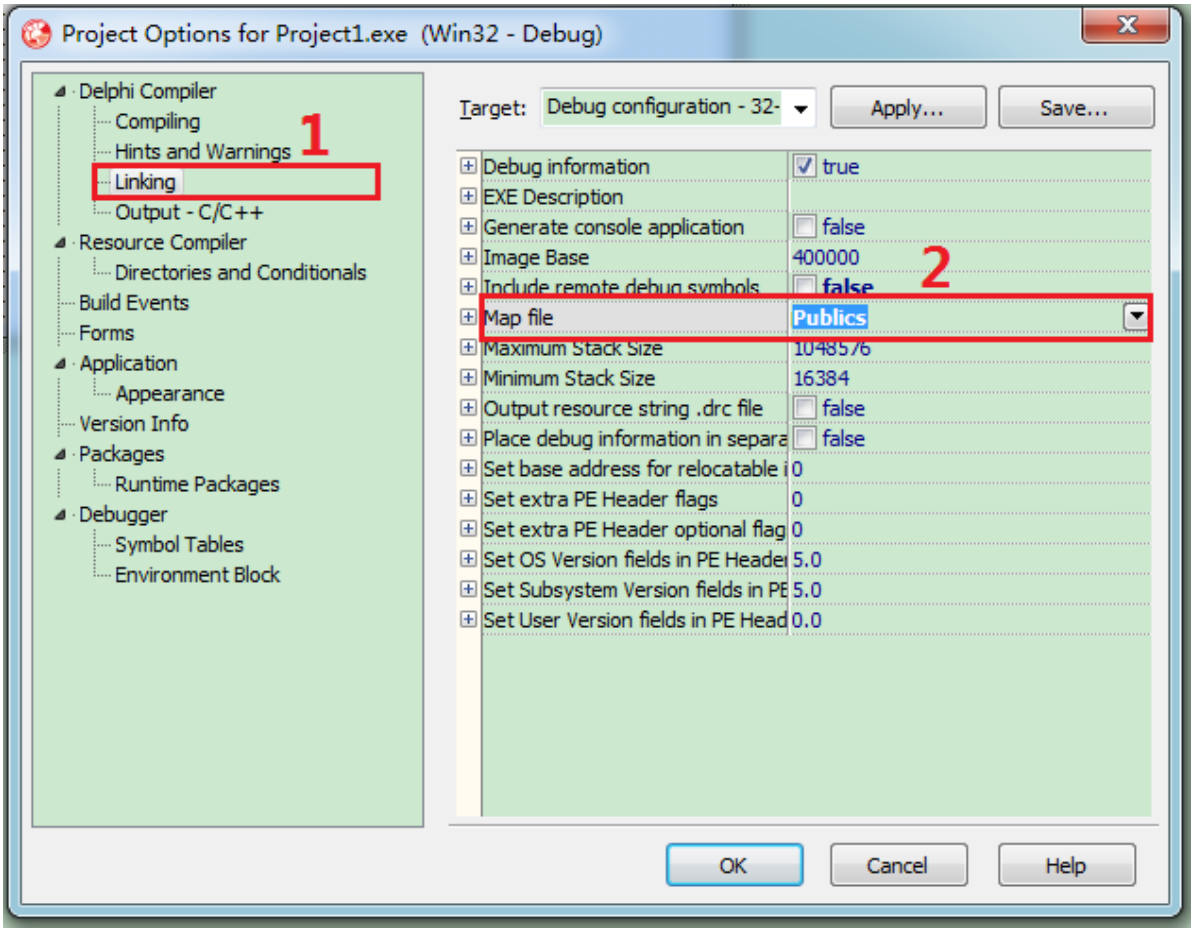
- 工程设置如下图：



使用 Delphi 生成 map 文件

- 工程设置如下图：


Project1 - Delphi XE2 - Unit1
File Edit Search View Refactor Project Run Component Tools
1
Structure
Form1
Object Inspector
Form1 TForm1
Properties Events
Action
Add to Project... Shift+F11
Remove from Project...
Add to Repository...
Languages
View Source
Add New Project...
Add Existing Project...
QA Metrics...
QA Audits...
Format Project Sources...
Modeling Support...
Compile Project1 Ctrl+F9
Build Project1 Shift+F9
Syntax Check Project1
Information for Project1
Compile All Projects
Build All Projects
Resources and Images...
Dependencies...
Project Page Options...
Deployment
2
Options... Shift+Ctrl+F11



使用 SDK 标签

SDK 工具包，包括头文件、静态库以及动态库，用户在编程的过程中将 SDK 标签静态载入到需要保护的函数当中，这样生成的可执行程序，在 Virbox Protector 加壳工具中就能够分析出 SDK 表示的函数，这样就能够找到用户的核心代码所在的位置。目前支持，VBProtectBegin（常规保护），VBVirtualizeBegin（虚拟化保护），VBMutateBegin（混淆化保护），VBSnippetBegin（碎片化代码保护）,VBProtectDecrypt(许可加解密)。[注意：SDK 标签能方便用户找到关键代码]

函数模块保护

注意事项

1. 只能静态加载，不支持动态加载dll（即 LoadLibrary 的方式）。
2. VBProtectBegin、VBVirtualizeBegin、VBSnippetBegin 以及 VBMutateBegin 等接口，传入的字符串参数，不能与其他函数共用。
3. 传入的字符串参数保证为 ANSII 码的形式，这样显示在界面上的函数名称才正确，否则就会显示为乱码。
4. 每个 Begin 对应一个 End，总是成对出现，并且一个函数里面不要出现多对 Begin+End。
5. 如果 SDK 表示的保护方式和工程文件中保存的保护方式冲突了，以工程文件中的保护方式为准。
6. Begin+End 锁定代码最好大于 3 行代码。（因为锁定的代码正汇编生成的指令小于 15 个字节，那么不会显示在加壳工具界面上）
7. SDK 动态库，分为 32 以及 64 位，在使用的时候要开发者根据要编译的程序位数进行加载对应的库。
8. 目前明确不支持的语言：易语言、Java 程序、Unity3d。
9. Begin/End 不支持嵌套使用。
10. VBProtectDecrypt 被加密的字符串或者是缓冲区的长度必须是 16 的倍数。

eg：char g_test_string[16] = {"test_decrypt"};

11. VBProtectDecrypt 传入缓冲区和传出缓冲区不能是同一个缓冲区。
12. VBProtectDecrypt 传入的缓冲区只能放在函数外，即全局变量。具体的使用参照 demo。
13. .Net 程序暂时不支持。

字符串加解密

1. 加密的字符串必须是常量。
2. 也可以使用 VBDecryptData 直接到数据加密，但数据和长度也必须是常量。
3. 字符串解密支持的写法有以下几种：

   1）直接字符串解密：

   ```
   VBDecryptStringA("test_string");
   ```

   2）局部静态变量：

   static const char g_string[] = "test_string";

   3）全局变量：

   char g_test_string[] = "test_string";

   const char g_test_string[] = "test_string";

   static const char g_test_string[] = "test_string";

   4）如果程序过于复杂可能会导致解析不出加密的数据，而在加壳时报错，使用 -fpic 或 -fpie 加上 -O2 编译的 32 位 Linux 程序此类情况比较明显。建议降低代码的复杂度。

   5）编译器可能会将相同的常量字符串合并为同一个，如果只加密了其中一个，会导致出错，如以下代码：

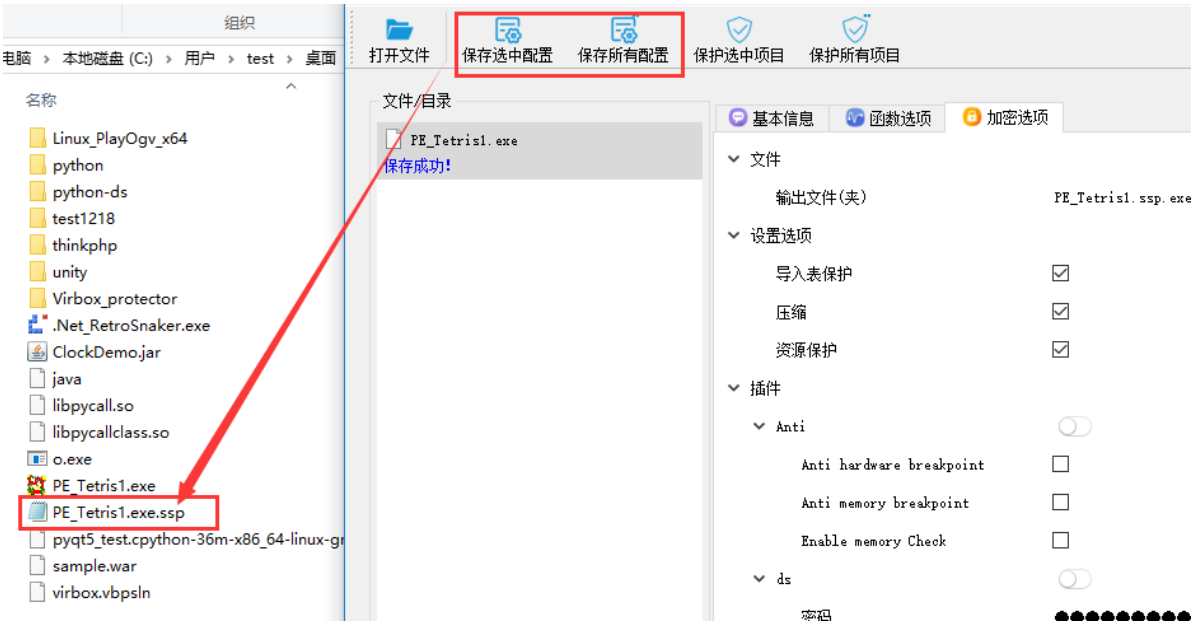
   ```
   const char* a = "test_string";
   const char* b = VBDecryptStringA("test_string");
   printf("a = %s, b = %s\n", a, b);
   ```

   这种情况，打印字符串a，可能会是乱码。

生成 .ssp 配置文件

手动生成 ssp 文件

将文件拖入到 Virbox Protector 工具中，手动更改信息后，点击“保存选中配置”或“保存所有配置”选项后，在和文件同一层目录下会生成一个 .ssp 配置文件。



使用命令行工具保护

Virbox Protector 命令参数解析

命令	描述(本地锁)	描述(云锁)	备注
filename	指准备保护的原文件		/
-u3d	指对 Unit3D 程序保护		
-o output	指保护后输出文件路径		

Linux命令行

1. 普通程序使用方法，以Linux平台程序为例：

   1）使用Virbox Protector界面工具生成配置文件（可选）

   - 若生成配置文件，可以在界面工具中可以对函数保护个数及保护类型进行选择；
   - 若不生成配置文件，则只会保护默认的入口函数；
   - 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息。

```
sense@sense:~/Desktop/virboxprotector_1.5.0.10808/bin$ ./virboxprotector_con
SenseShield Virbox [version: 1.5.0.10808]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


VirboxProtector_con <filename> [-o output]
 -u3d                          : protect for unity3d.
 -o output              : output file name.
 -?                     : show help information.
```

   2）针对不同平台的程序，独立壳对其许可的限制不同，需要联系**深思销售**获取许可。

   命令：VirboxProtector_con的路径 需要被保护的程序路径 -o 输出文件的路径

   - 没有获取许可，使用Virbox Protector保护将提示“ Can not find the license”，如图所示：

```
sense@sense:~/Desktop/virboxprotector_1.5.0.10808/bin$ ./virboxprotector_con '/h
ome/sense/Desktop/virbox_test/cb_bytes_test' -o '/home/sense/Desktop/virbox_test
/cb_bytes_test.ssp.vp'
SenseShield Virbox [version: 1.5.0.10808]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


loading cb_bytes_test ...
Error (13000020): Can not find the license.
```

- 获取许可后，使用Virbox Protector保护成功，如图所示：

```
sense@sense:~/Desktop/virboxprotector_1.5.0.10808/bin$ ./virboxprotector_con '/h
ome/sense/Desktop/virbox_test/cb_bytes_test' -o '/home/sense/Desktop/virbox_test
/cb_bytes_test.ssp.vp'
SenseShield Virbox [version: 1.5.0.10808]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


loading cb_bytes_test ...
link...
save as cb_bytes_test.ssp.vp ...
Succeed.
```

Windows命令行

1. 普通程序使用方法，以 Linux 平台程序为例：

   1）使用 Virbox Protector 界面工具生成配置文件（可选）；

   - 若生成配置文件，可以在界面工具中可以对函数保护个数及保护类型进行选择；
   - 若不生成配置文件，则只会保护默认的入口函数。

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

```
C:\Users\test\Desktop\virboxprotector_standalone_1.4.2.10236_windows_x64\bin>virboxprotector_con.exe
SenseShield Virbox [version: 1.4.2.10236]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


VirboxProtector_con <filename> [-o output]
 -u3d                          : protect for unity3d.
 -o output              : output file name.
 -?                     : show help information.
```

   3）针对不同平台的程序，独立壳对其许可的限制不同，需要联系**深思数盾销售人员**获取许可；

> 命令：VirboxProtector_con.exe的路径 需要被保护的程序路径 -o 输出文件的路径

   - 没有获取许可，使用Virbox Protector保护将提示“ Can not find the license”，如图所示：

```
C:\Users\test\Desktop\virboxprotector_standalone_1.4.2.10236_windows_x64\bin>virboxprotector_con.exe C:\User
s\test\Desktop\sample\abstract-class32 -o C:\Users\test\Desktop\sample\abstract-class32.ssp.vp
SenseShield Virbox [version: 1.4.2.10236]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


loading abstract-class32 ...
Error (13000020): Can not find the license.
```

   - 获取许可后，使用Virbox Protector保护成功，如图所示：

```
C:\Users\test\Desktop\virboxprotector_standalone_1.4.2.10236_windows_x64\bin>virboxprotector_con.exe C:\User
s\test\Desktop\sample\cb_bytes_test -o C:\Users\test\Desktop\sample\cb_bytes_test.vp
SenseShield Virbox [version: 1.4.2.10236]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


loading cb_bytes_test ...
link...
save as cb_bytes_test.vp ...
Succeed.
```

.NET 程序选项

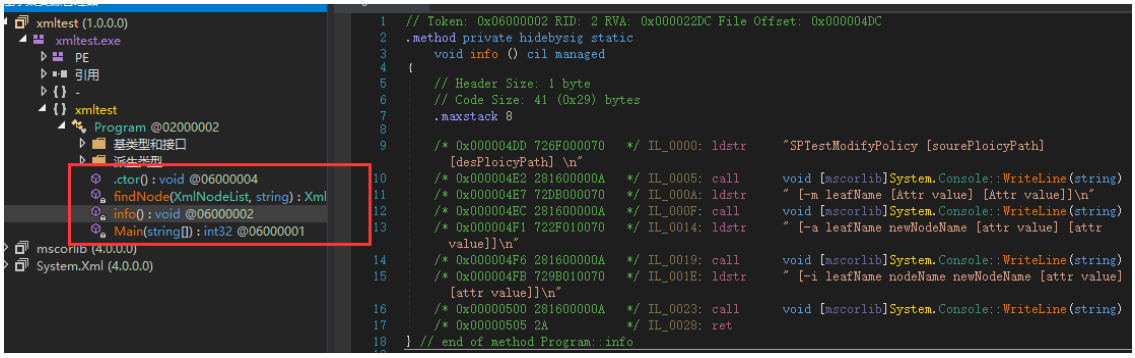
基础保护

名称混淆（.NET）

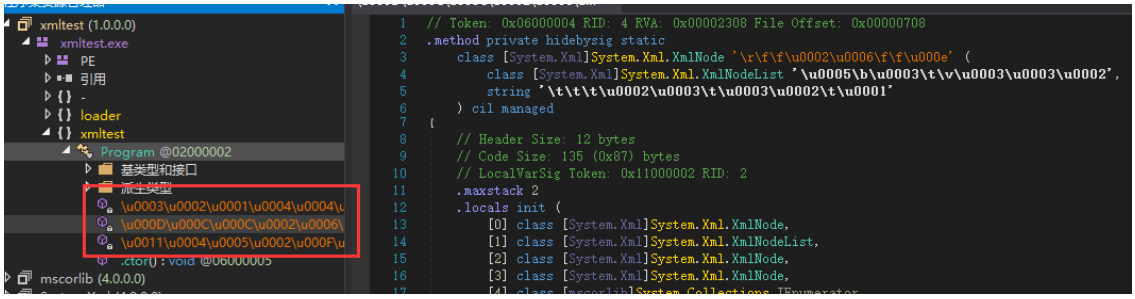
将 .net 的方法名类名使用随机字符串重新命名，导出和外部的名称不会改变。

保护效果图

- 保护前，如图所示：



- 保护后，如图所示：

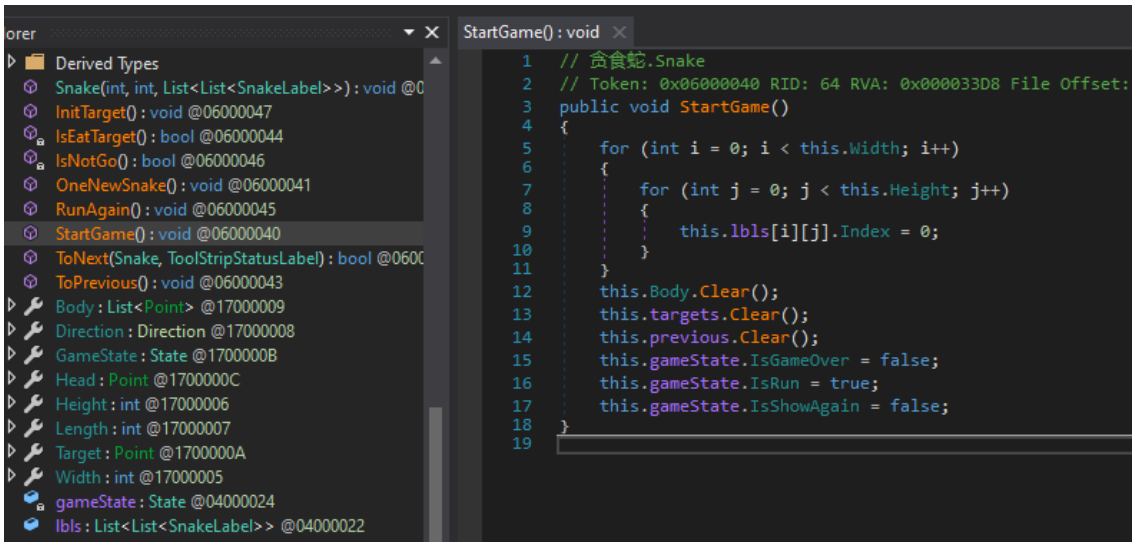


压缩

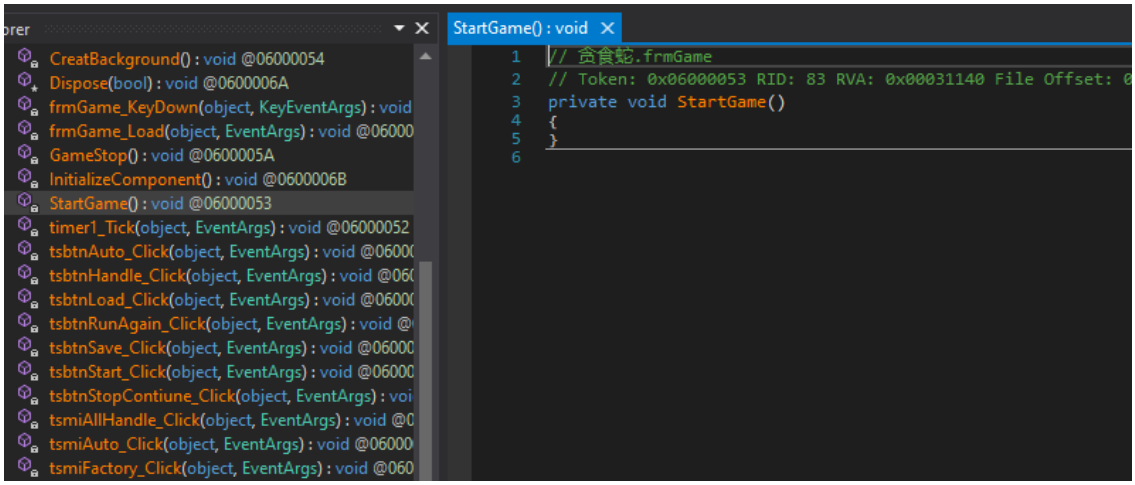
.NET 的压缩功能主要为 .NET 程序提供了整体保护的效果，可以防止 .NET 程序中的方法被 DnSpy，ILSpy，.NET Reflector 等工具反编译，使用 IL 壳代码，提升了兼容性。

保护效果图

- 保护前，如图所示：



- 保护后，如图所示：



JIT加密

描述

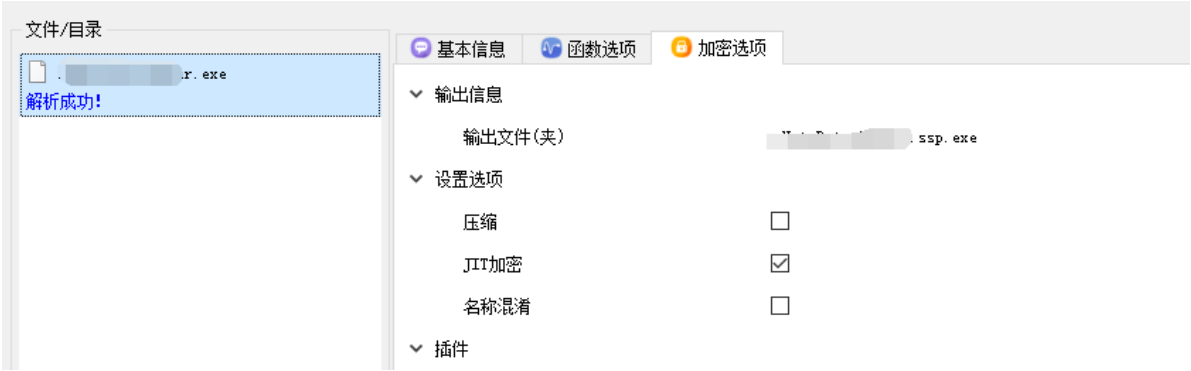
1、.NET JIT 加密，是将 .NET 所有方法的 IL 指令经过加密，仅在 .NET 虚拟机进行 JIT 编译阶段才解密，可以防止静态反编译，也能防止 IL 代码在内存被 Dump。

2、JIT 加密默认对所有方法进行加密，更进一步提升保护后代码的安全性。

3、JIT加密支持了继承、事件、反射、递归调用等代码加密无法支持的写法。

使用方法

将.NET程序拖入到Virbox Protector工具中，加密选项就会显示“JIT加密”选项，勾选上此选项，对程序进行保护，则保护后的程序将拥有此功能的效果。



去除强签名

1. 强名称(StrongName)使.NET提供的一种验证机制, 主要包括标识版本和标识原作者。
2. 强名称可以用来帮助用户验证自己得到的程序是否为原作者所写切没有被修改(例如添加恶意代码), 跟自校验有点类似。
3. 因此添加了强名称的程序加壳时要去除强名称, 并在加壳后重新添加强名称。

函数级保护

代码加密（.NET）

原理

代码加密是使用动态代码技术，将原始方法字节码加密，执行时才将方法解密并执行的保护方式。

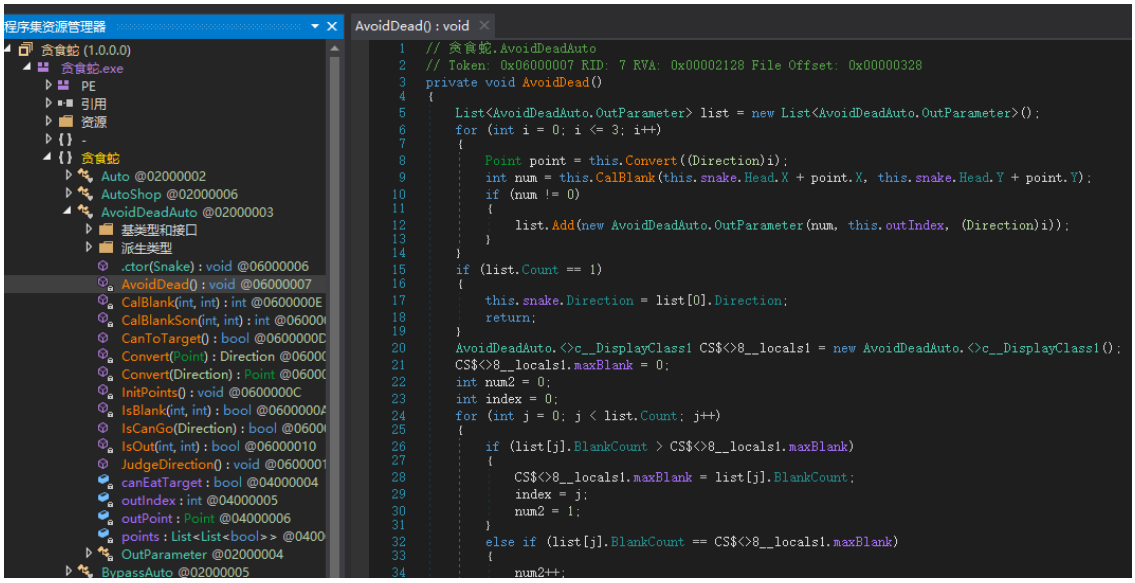
功能

防脱壳，防止直接 Dump。

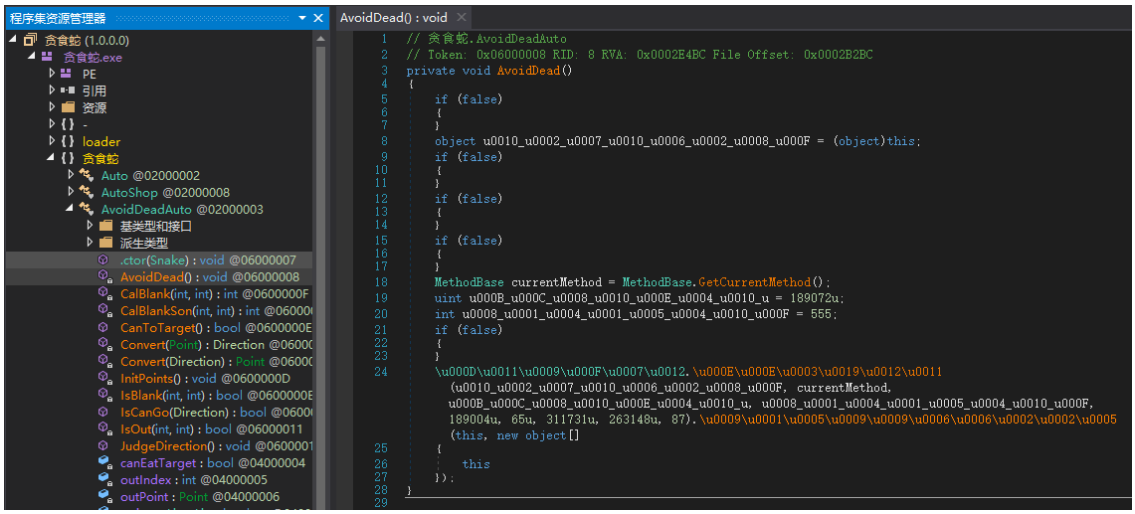
- 优点
    1. 运行效率高，几乎没有性能损失。
- 缺点
    1. 方法执行后会解密，解密之后容易被分析。

保护效果图

- 保护前，如图所示：



- 保护后，如图所示：



代码加密不支持类型

针对C#程序选择函数的保护方式为代码加密时，加壳时提示“部分被保护函数设置了其不支持的保护方式，请前往函数选择界面更改保护方式。[0xA000A000]”的情况，以下列出代码中不支持的写法：

1. 值类型（及其继承类）的非静态方法 System.ValueType
2. 泛型方法暂不支持
3. C++ .net不支持
4. 递归调用不支持
5. 可变参数不支持
6. 默认参数不支持

代码混淆

原理

代码混淆亦称花指令，是将计算机程序的代码，转换成一种功能上等价，但是难于阅读和理解的形式。

Virbox Protector 支持对 x86/arm/.net il 系列指令进行混淆。

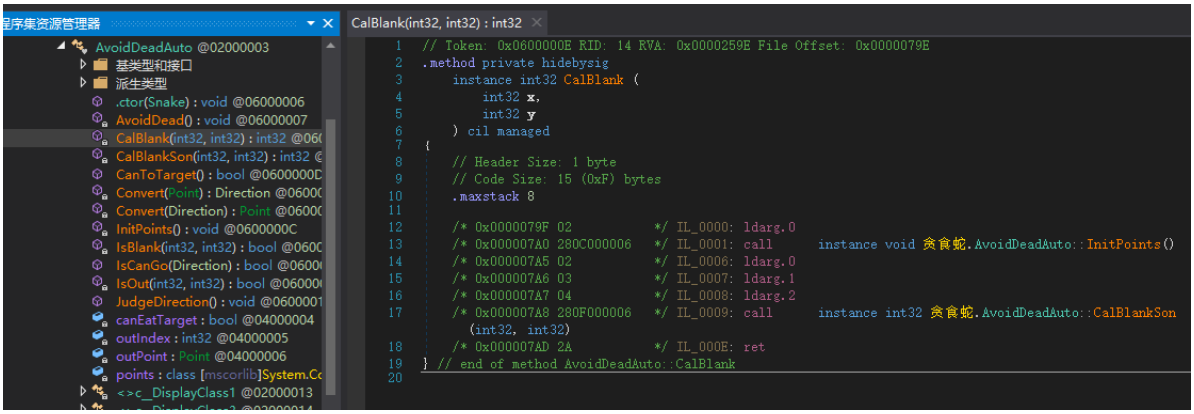
功能

扰乱原始指令，防止静态分析。

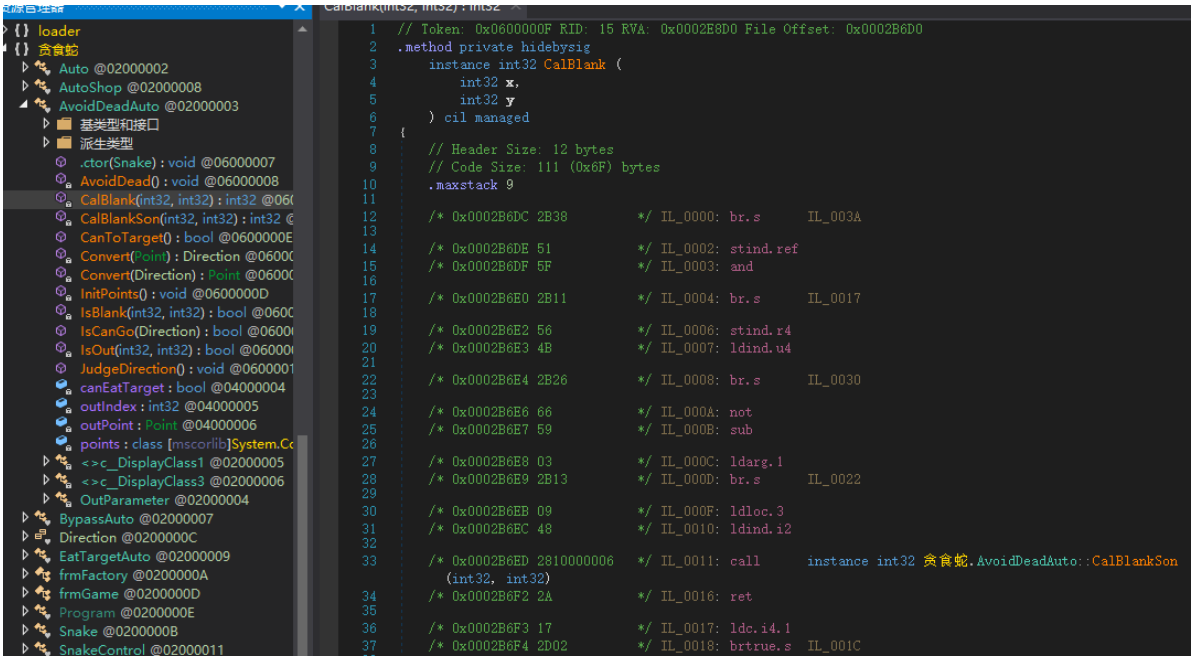
- **优点**
    1. 防反编译。
- **缺点**
    1. 运行效率略有损失，保护强度不高。

保护效果图

- 保护前，如图所示：



- 保护后，如图所示：



NET标签保护

.NET程序支持代码加密和代码混淆两种函数保护方式，代码中设置Virbox命名空间，并在代码中引用，程序编译成功后，将编译好的程序拖入到加壳工具界面，界面会显示代码中设置的函数保护方式：

代码书写方式：

```
//命名
namespace Virbox{
    //代码混淆
    class Mutate : System.Attribute
    {
    }
    //代码加密
    class Encrypt: System.Attribute
    {
    }
}
public class main
{
    [Virbox.Mutate]//代码混淆
    public static void test1(string[] args)
    {
        System.Console.WriteLine("hello Virbox.Mutate!");
    }
    [Virbox.Encrypt]//代码加密
    public static void test2(string[] args)
    {
        System.Console.WriteLine("hello Virbox.Encrypt!");
    }

    public static void Main(string[] args)
    {
        test1(args);
        test2(args);
    }
}
```

Unity3D 程序保护

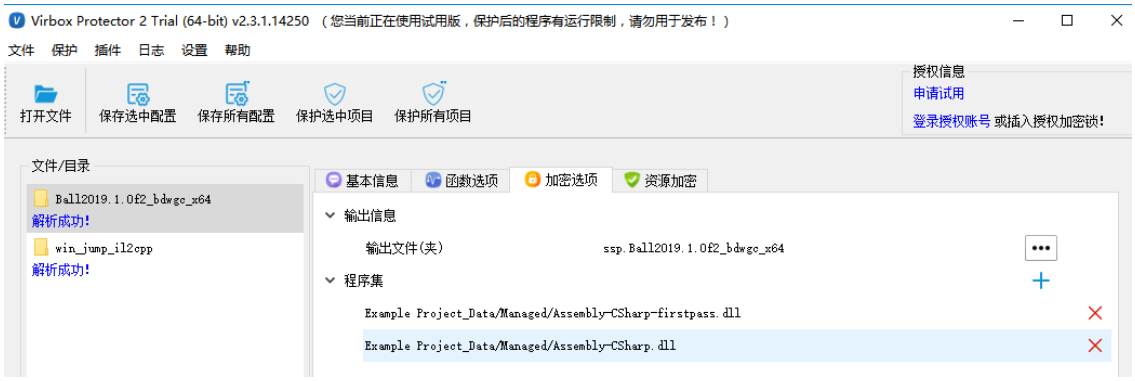
使用流程

Windows、Linux和macOS平台

界面使用流程

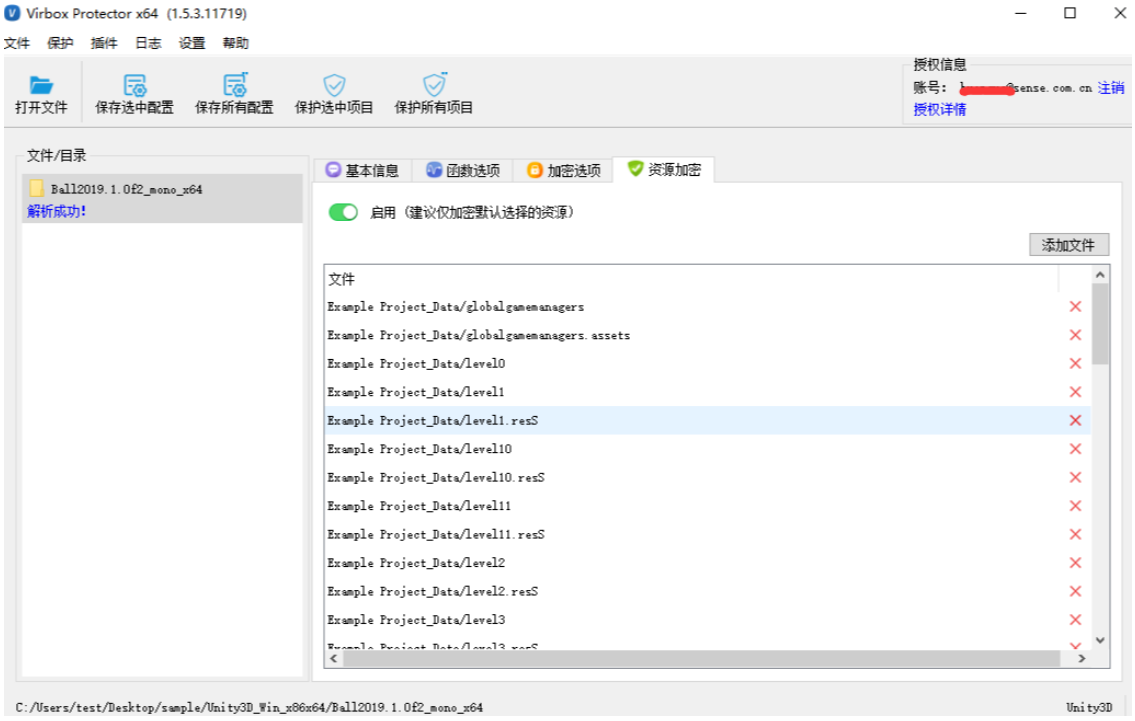
1. 针对Windows、Linux和macOS系统上的Unity3D程序，将Unity3D目录直接拖入到加壳工具中;
2. 加密选项设置：

   mono格式：加密选项处默认会加载程序中的Assembly-CSharp.dll和Assembly-CSharp-firstpass.dll文件，也可以添加Managed目录下自主开发的C# 程序集;



il2cpp格式：加密选项处默认对元数据（global-metadata.dat）进行保护；

3. 可以直接在资源加密选项处点开启用按钮，可以对Unity3D中的资源文件进行加密保护;【注】资源加密选项建议仅加密默认选择的资源。



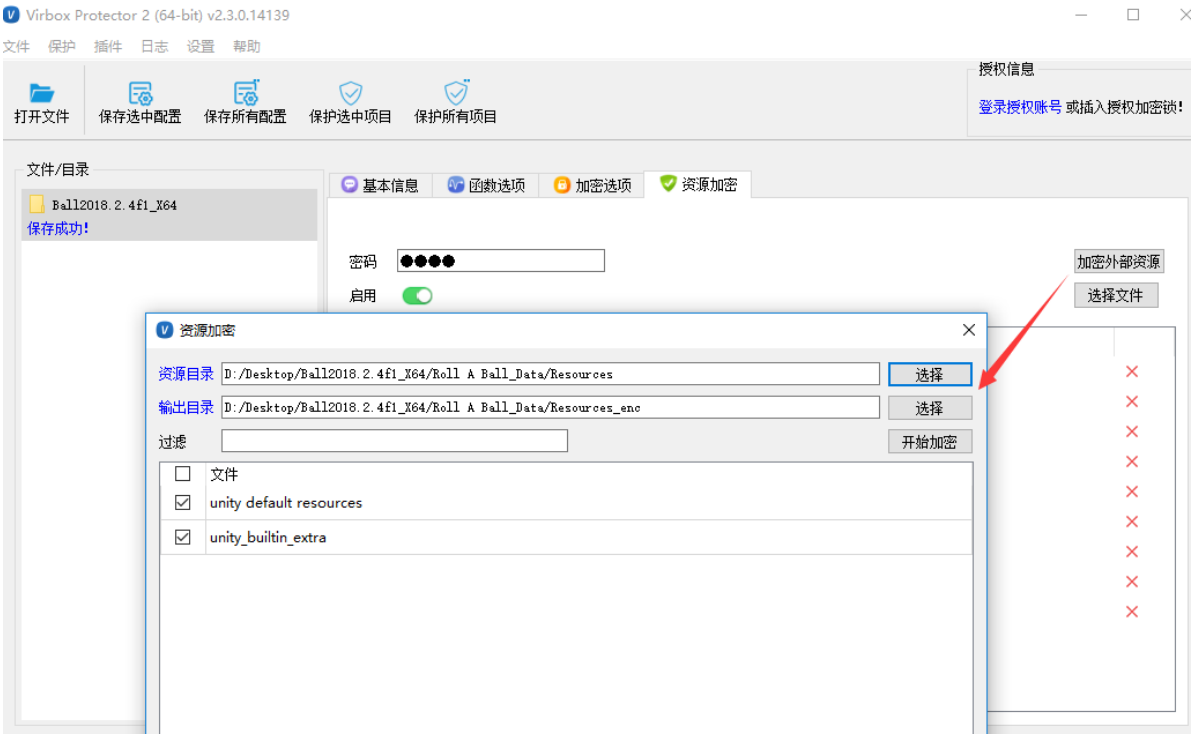
4. 加壳成功后会生成ssp.Unity3D目录名，可以直接运行。

热更新

1、为了方便加密保护后Unity3d资源的更新，“加密外部资源”功能可以对资源单独进行加密，然后将加密生成后的资源去进行更新替换。

【注意】

- 密码保持一致，即加密项目和加密外部资源的密码一样；
- 目前只支持Windows 和Linux unity3D mono格式的程序。



命令行保护

unity目录

1. 使用Virbox Protector界面工具生成配置文件；
2. 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息；
3. 命令如下：

```
virboxprotector_con  <需要被保护的目录> -o <输出文件的目录>
```

资源更新目录

1. 使用Virbox Protector界面工具生成配置文件；
2. 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息；
3. 命令如下：

```
virboxprotector_con  -u3dres <需要资源更新的目录> -x <配置文件> -o <输出资源保护的目录>
```

以Linux Unity3D为例：

1. 使用Virbox Protector界面工具生成配置文件（可选）；
2. 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息；
3. 针对不同平台的程序，独立壳对其许可的限制不同，需要联系**深思销售**获取许可；

   命令：VirboxProtector_con的路径 需要被保护的程序全路径 -u3d -o 输出文件的路径；

1）没有获取许可，使用Virbox Protector 保护将提示“ Can not find the license”，如图所示：

```
sense@sense:~/Desktop/virboxprotector_1.5.0.10808/bin$ ./virboxprotector_con '/h
ome/sense/Desktop/Particles2018.1.9f1' -u3d -o '/home/sense/Desktop/ssp.Particle
s2018.1.9f1'
SenseShield Virbox [version: 1.5.0.10808]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


protect unity3d Particles2018.1.9f1 ...
Error (13000020): Can not find the license.
```

2）获取许可后，使用Virbox Protector保护成功，如图所示：

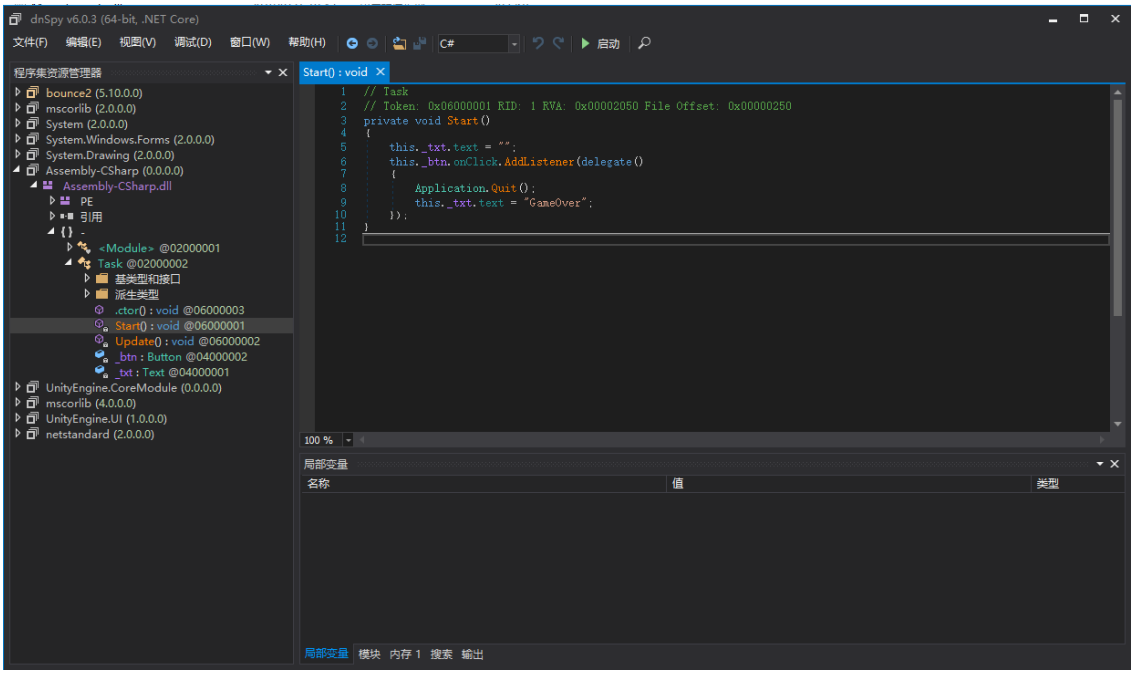
```
sense@sense:~/Desktop/virboxprotector_1.5.0.10808/bin$ ./virboxprotector_con '/h
ome/sense/Desktop/Particles2018.1.9f1' -u3d -o '/home/sense/Desktop/ssp.Particle
s2018.1.9f1'
SenseShield Virbox [version: 1.5.0.10808]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


protect unity3d Particles2018.1.9f1 ...
Succeed.
```

保护原理

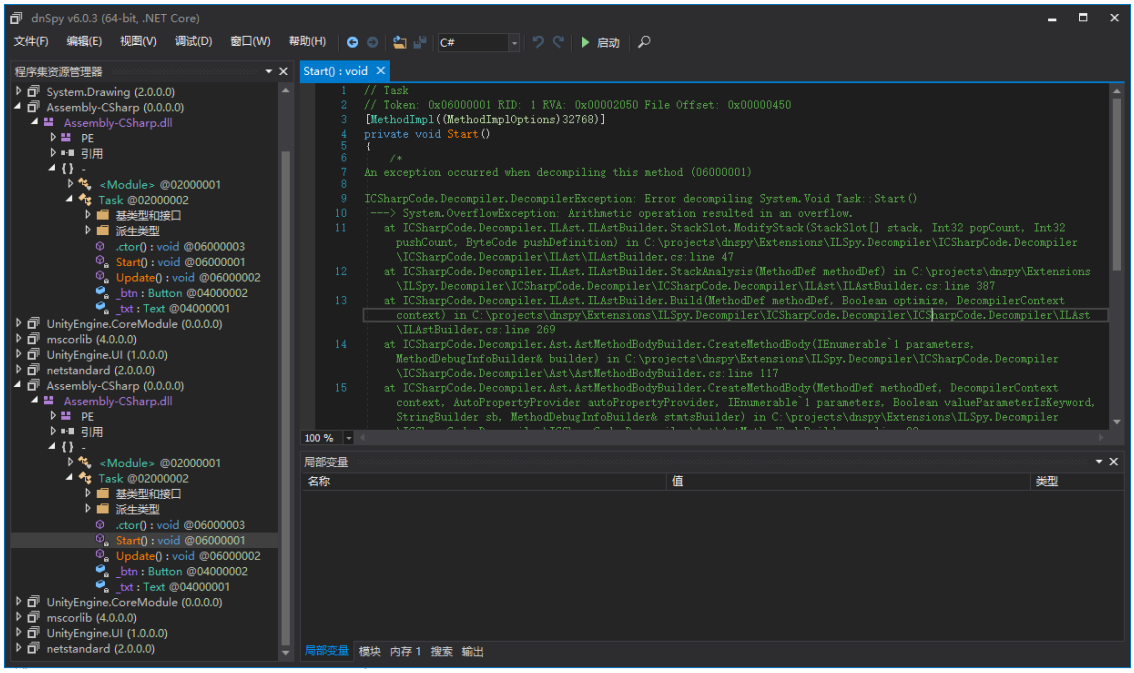
1. 解析 Assembly-CSharp.dll 脚本文件，将 function 转换成 IL 代码;
2. 将 IL 代码进行加密，其中密钥为每次随机生成，保持到脚本文件中;
3. 如果使用 LM 版本的壳，加解密使用锁内密钥;
4. 链接重新生成 Assembly-CSharp.dll 脚本文件，此时所有代码已经被加密;
5. 对 Unity3D 的 .NET 运行时库 mono 进行处理，定位到解析 .NET 方法的函数并进行 hook;
6. 插入 hook 代码对 Assembly-CSharp.dll 的方法解密，重新编译生成新的 mono 并替换原始动态库。

保护效果图

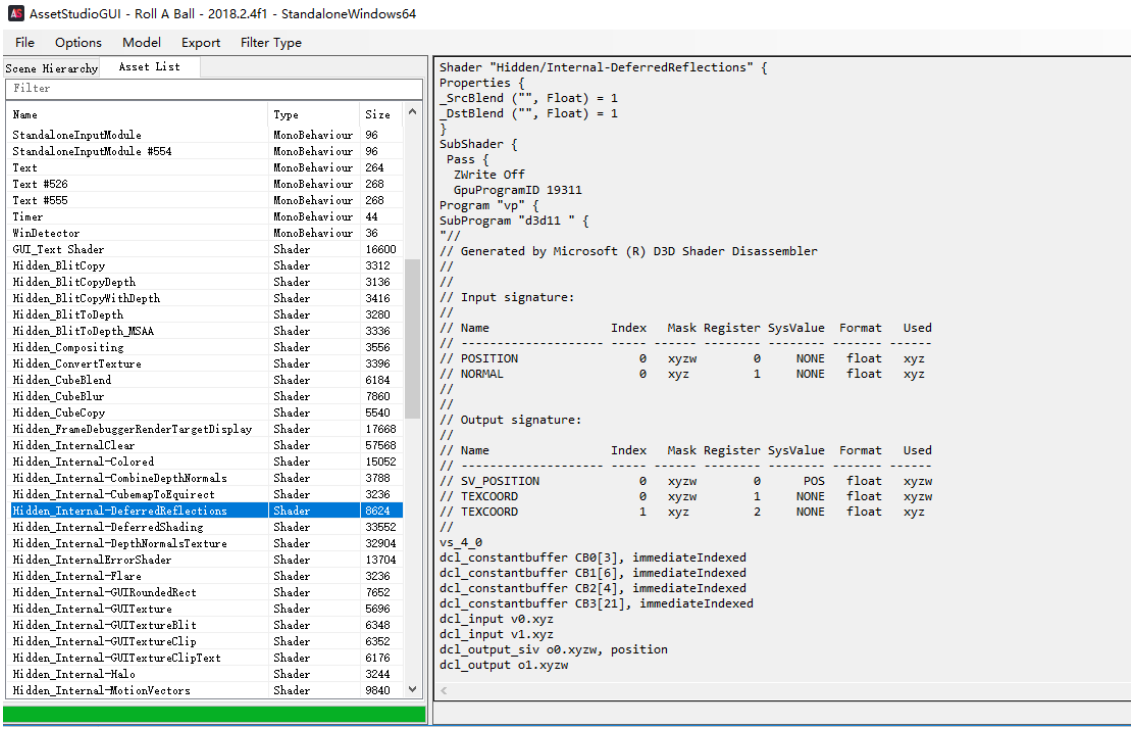
1. 下图为 Assembly-CSharp*.dll 脚本文件的保护效果图。
    - 保护前，如图所示：



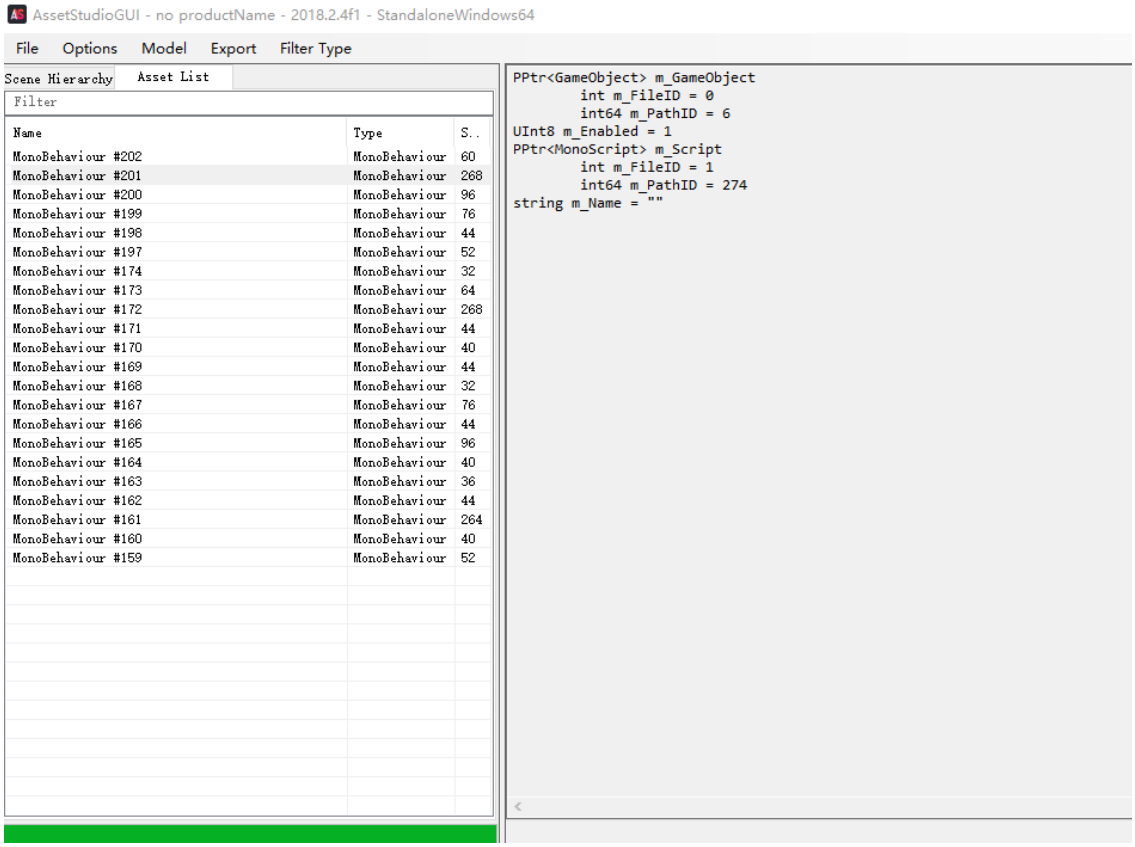
    - 保护后，如图所示：



2. Unity3D资源文件的保护效果，可以使用反编译工具(例：AssetStudio)对Unity3D资源进行反编译查看一下效果。
    - 原始Unity3D资源文件反编译效果，如图所示：



    - 对resS、assets和resource资源文件进行保护后的反编译效果，如图所示：



Android 平台

界面使用流程

1、直接将目标APK或AAB程序拖入到加壳工具界面；

【注】APK或AAB程序已经不缺分mono和il2cpp格式加壳方式了，目前是统一对APK或AAB程序加壳。

2、在加密选项处进行设置：

- 勾选内存校验，壳代码会对每个要校验的内存块进行校验，以验证其完整性，如果校验失败，则会清场退出。
- 勾选Metadata加密，是加密 il2cpp 的 global-meta-data 文件，并对内部结构进行混淆处理，防止运行时在内存中直接解析。
- 勾选Metadata名称混淆，是 il2cpp 的 global-meta-data 文件中的方法名进行混淆处理。

- 勾选反调试按钮，则使用调试工具调试时，程序会直接退出。
- 勾选签名校验按钮，且进行签名设置，输入自己的keystore文件和密码（密钥别名和密钥密码选填）。
- 勾选反注入，可以防止其它进程对 APK或AAB 进程附加调试或注入。
- 勾选模拟器检测，可以防止程序在“夜神”“雷电”等模拟器中运行。
- 勾选root检测，可以防止程序在root过后的手机上运行。
- 勾选多开检测，可以防止程序多开分身。

3、可以直接在资源加密选项处点开启用按钮，可以对Unity3D中的资源文件进行加密保护；

4、保护所选项目按钮，等待完成即可。

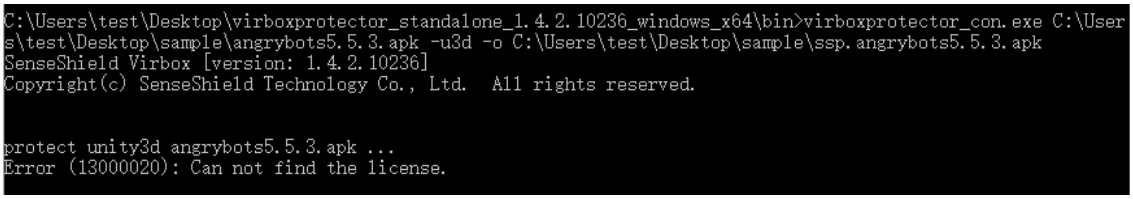
命令行保护

Unity3D 作为一个特殊的程序，和普通程序的保护方式不同，针对 Android平台的 Unity3D，需要对 Android Unity3D 的 apk或aab进行保护。

1. 使用 Virbox Protector 界面工具生成配置文件（可选）；
2. 打开终端窗口，进入到“virboxprotector_con.exe”所在的路径，直接输入“virboxprotector_con.exe”运行可查看帮助信息；
3. 针对不同平台的程序，独立壳对其许可的限制不同，需要联系**深思销售**获取许可；

```
virboxprotector_con  <需要被保护的apk/aab> -o <输出文件的apk/aab>
```

1）没有获取许可，使用 Virbox Protector 保护将提示“ Can not find the license”，如图所示：



2）获取许可后，使用 Virbox Protector 保护成功，如图所示：

iOS平台

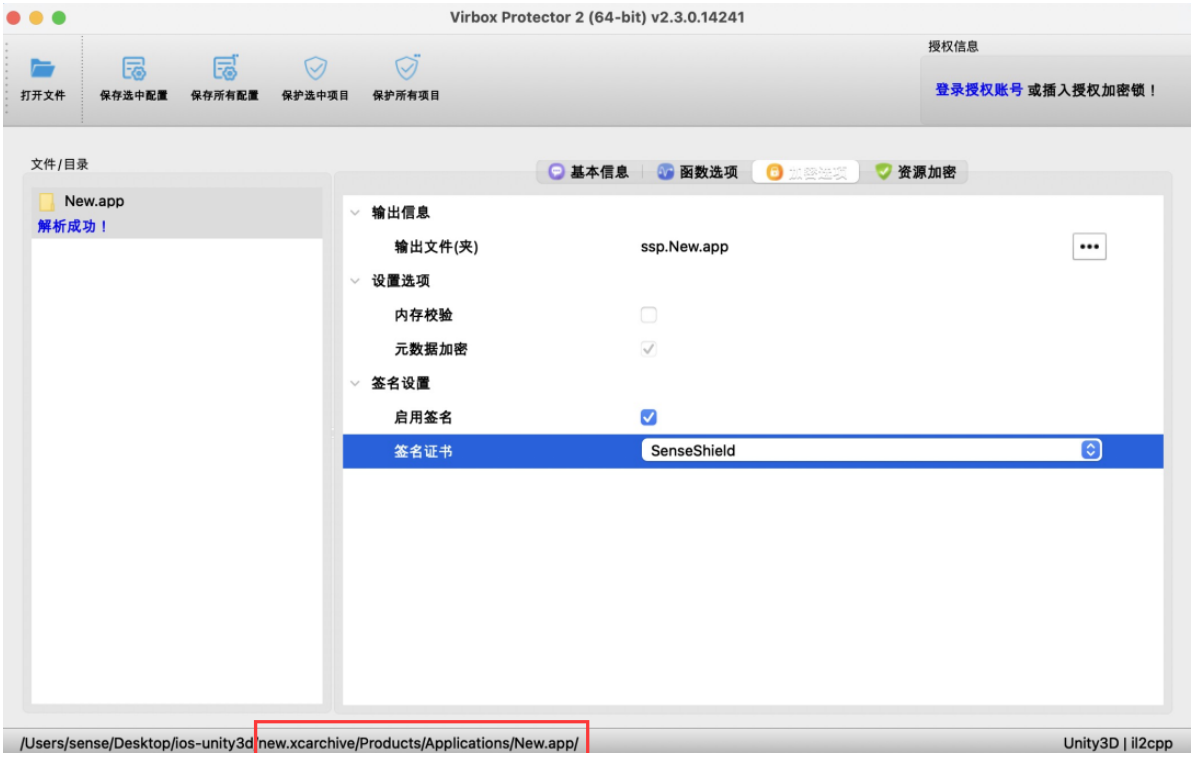
界面使用流程

编译xcarchive包

1、使用Xcode 编译xcarchive包；

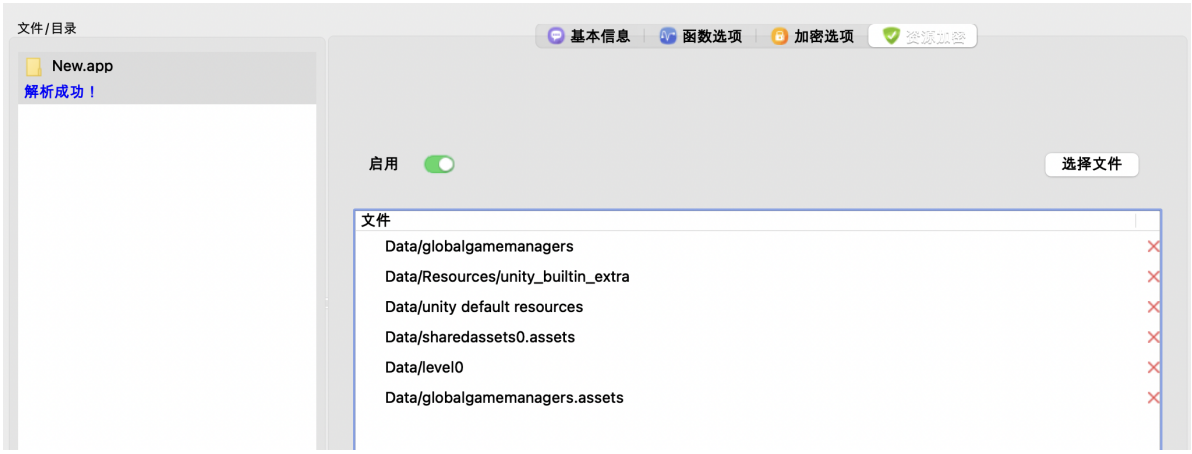
VBP加壳

2、然后将xcarchive\Products\Applications里的app程序直接拖入到加壳工具界面；



5、勾选启用签名，可以对保护后的程序自动进行签名；

6、点击“启用”按钮，可以对资源文件进行保护；



7、点击“保护选中项目”或“保护所有项目”对目标app进行保护。

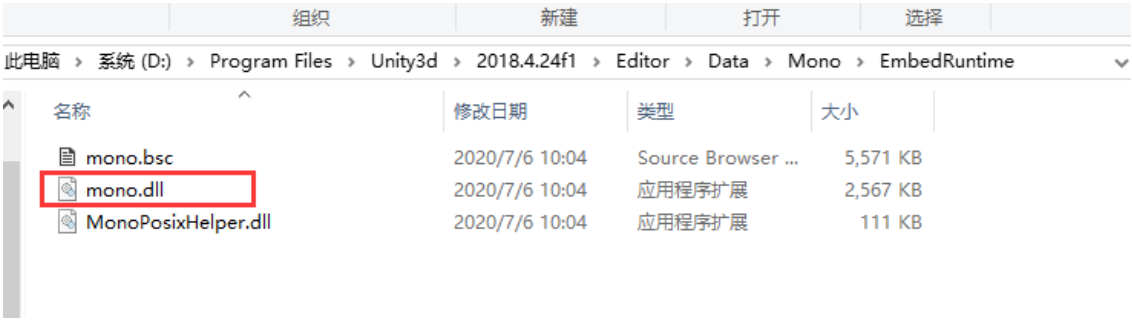
命令行保护

1. 使用Virbox Protector界面工具生成配置文件(若无配置文件，则命令行加壳后的app默认不签名)；
2. 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息；
3. 命令如下：

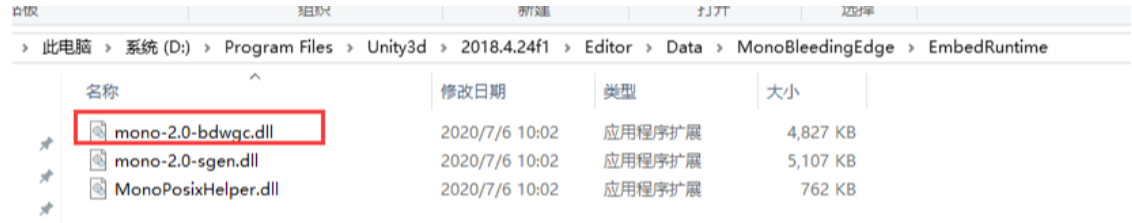
```
virboxprotector_con  <需要被保护的app> -o <输出文件的app>
```

Unity3D调用Net dll插件

1. 首先查找Unity编译器中mono.dll或mono-2.0-bdwgc.dll的位置。
    - mono.dll一般在.\Editor\Data\Mono\EmbedRuntime目录下

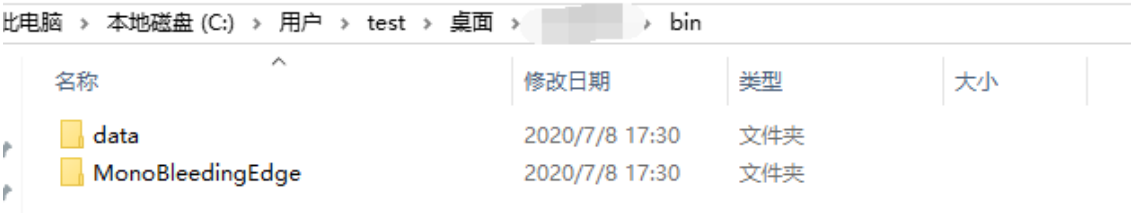


- mono-2.0-bdwgc.dll一般在.\Editor\Data\MonoBleedingEdge\EmbedRuntime

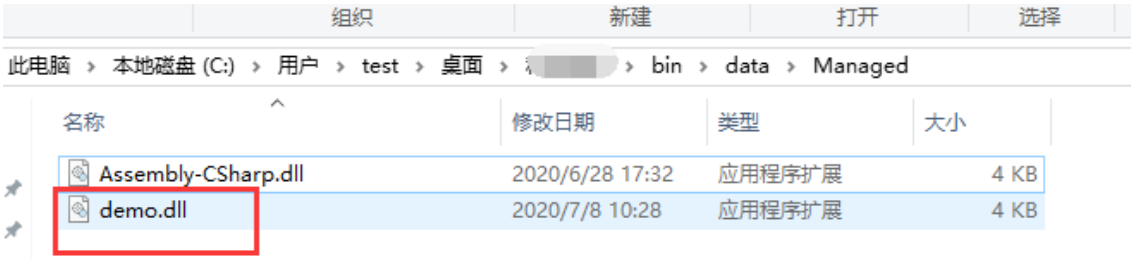


2. 可以创建了一个假的Unity3d程序目录，目录和普通Unity3d目录结构一样，主要是为了加壳工具可以识别并成功加壳。**以下为mono-2.0-bdwgc为例**

   1）将mono-2.0-bdwgc.dll放入bin\MonoBleedingEdge\EmbedRuntime\目录下;

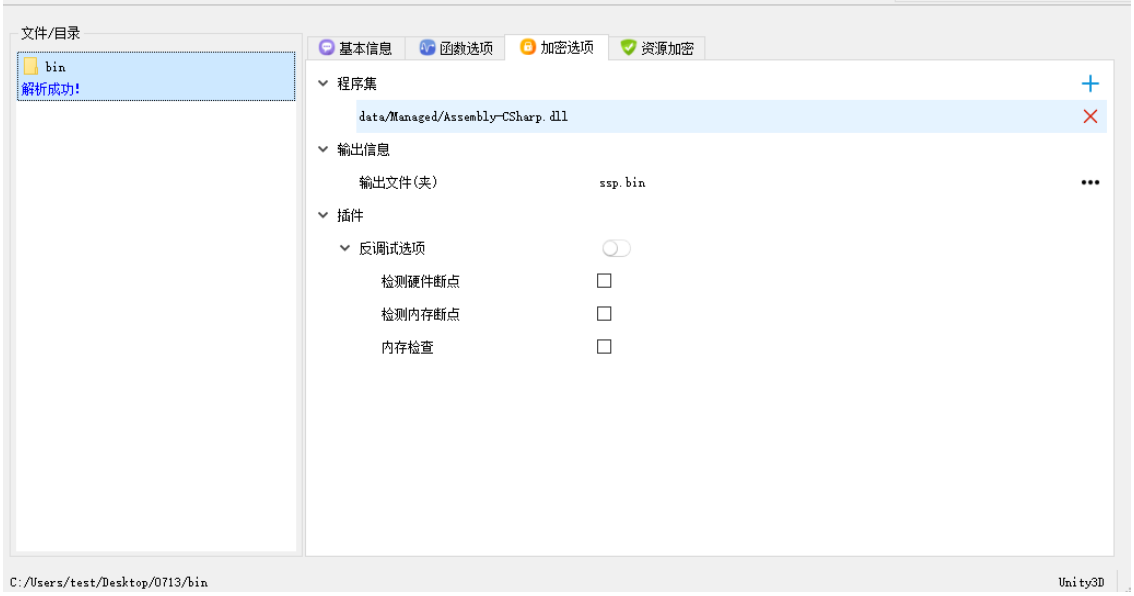


   2）将dll插件(比如demo.dll)放入到bin\data\Managed目录下;

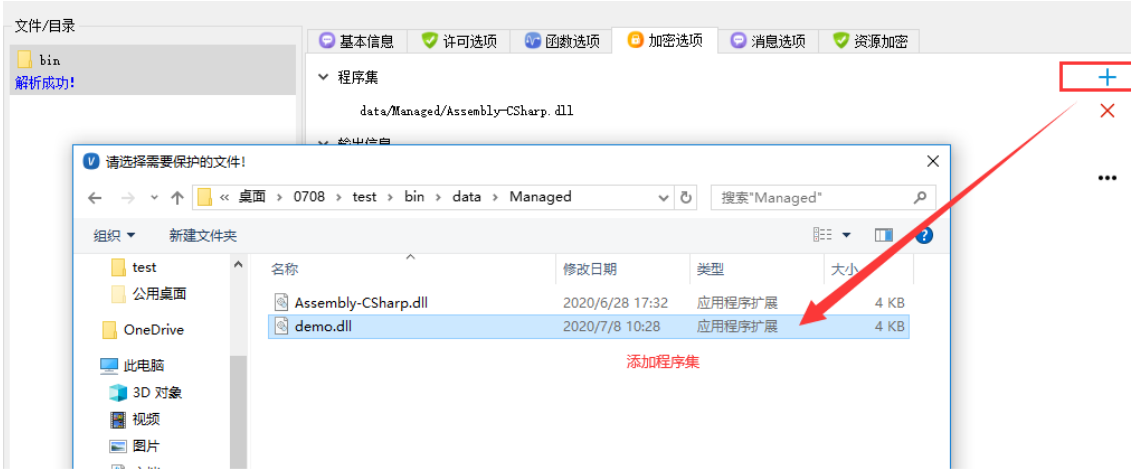


3. 将bin目录拖入到加壳工具中。

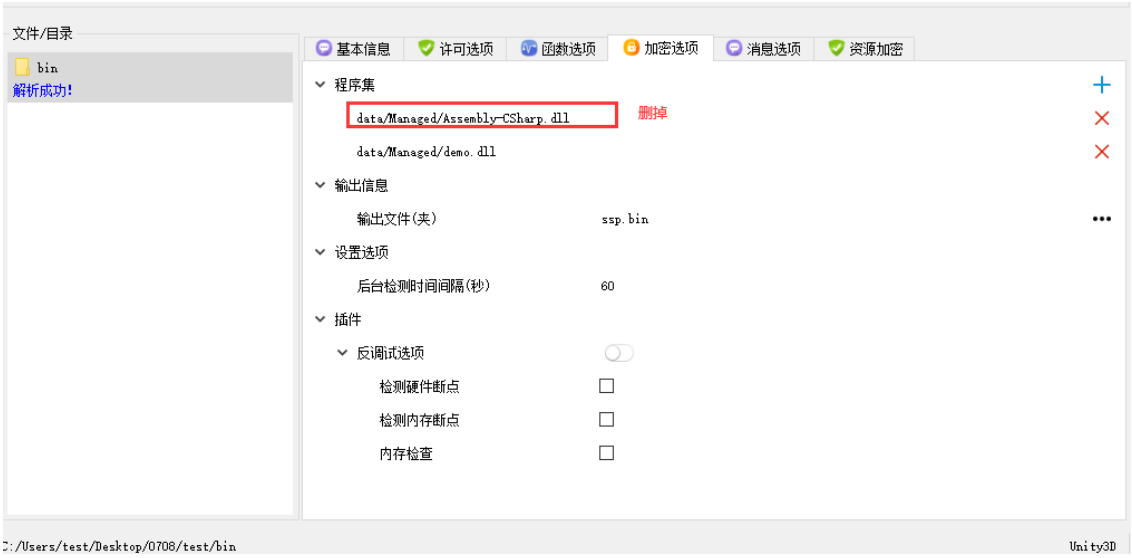
   1）设置相关配置。



2）进入到加密选项，点击添加程序集。

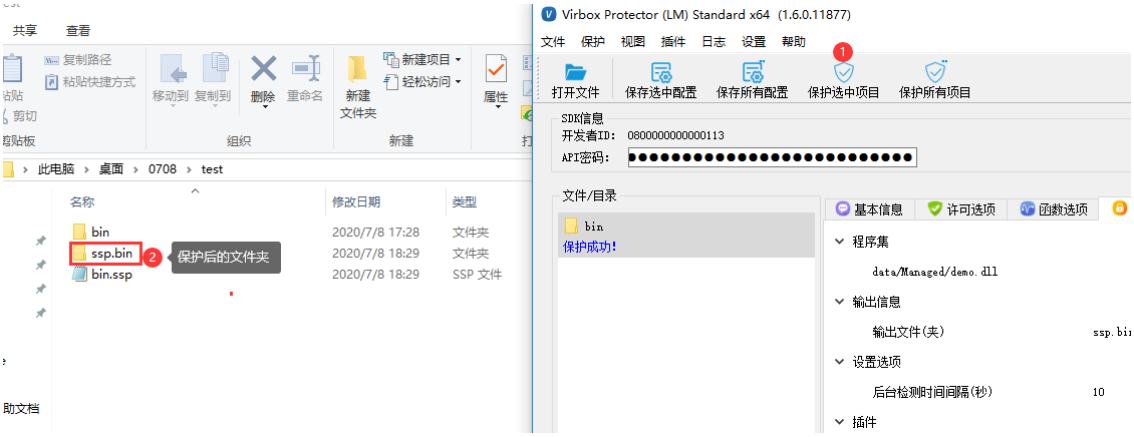


3）添加成功后，将多余的程序集给删除掉，保留自己的插件dll。



4）点击“保护选择项目”，进行加壳，加壳成功后会生成ssp.bin。

【注】资源加密选项不能启用。



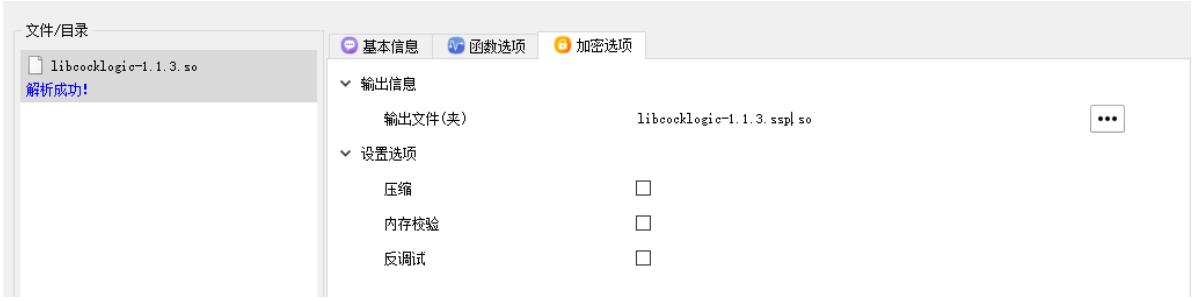
4. 进入到ssp.bin目录下，将ssp.bin\MonoBleedingEdge\EmbedRuntime下的mono-2.0-bdwgc.dll拷贝到Editor\Data\MonoBleedingEdge\EmbedRuntime目录下(原始文件备份)，将ssp.bin\data\Managed目录下的demo.dll放入到项目中即可。

Android程序保护

so库保护

1、针对普通apk程序，需要将apk解压，然后对文件夹lib目录下的so库使用加壳工具进行保护，如图所示:

2、可以对so库选择"压缩"、"内存校验"、"反调试"功能，函数选项可以选择自己想要保护的函数。



3、保护完成后，放入到apk中即可。

APK或AAB保护

反调试功能，有效抵挡动态调试，可以避免逆向工具分析获取源码 ，签名校验功能，可以防止二次打包签名，可以对Android APK或AAB里关键代码、核心逻辑进行加密保护。

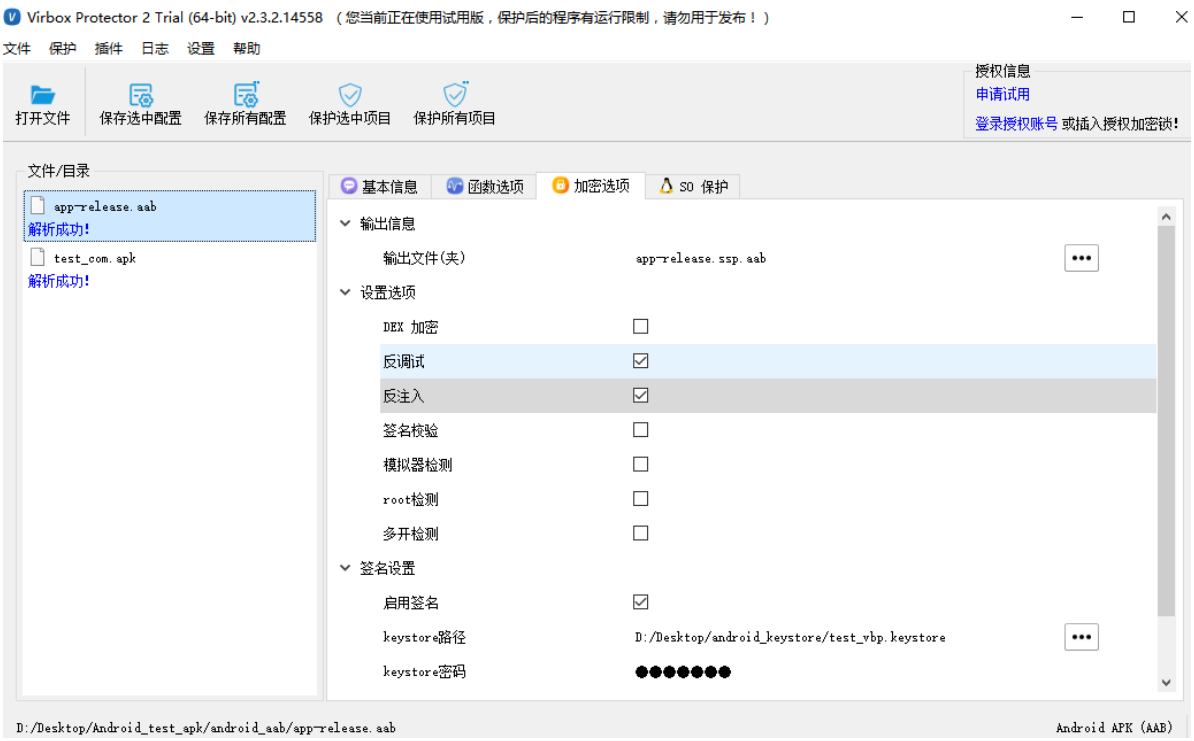
使用流程

1、将Android APK或AAB直接拖入到加壳工具中。

2、函数选项处可以对dex里的函数进行虚拟化保护。

3、在加密选项处进行设置：

- dex加密是对 DEX 文件整体压缩加密（若在Google Play上架，不建议勾选dex加密，建议选择虚拟化方式保护dex文件里的函数）
- 勾选反调试按钮，则使用调试工具调试时，程序会直接退出。
- 勾选签名校验按钮，且进行签名设置，输入自己的keystore文件和密码（密钥别名和密钥密码选填）
- 勾选反注入，可以防止其它进程对 APK或AAB 进程附加调试或注入。
- 勾选模拟器检测，可以防止程序在“夜神”、“雷电”、"AVD"等模拟器中运行。
- 勾选root检测，可以防止程序在root过后的手机上运行。
- 勾选多开检测，可以防止程序多开分身。

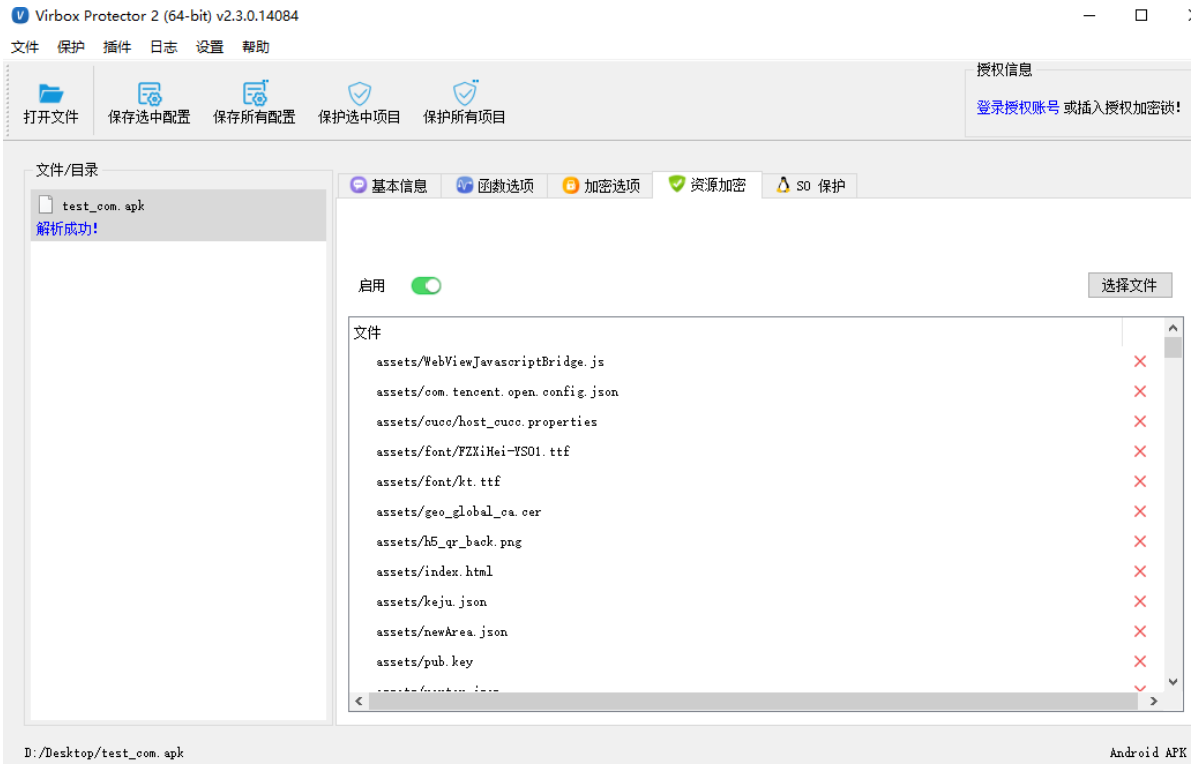


3、资源加密

加密APK/AAB中的assets下的文件，支持图片、配置、脚本等文件类型。

如果指定密码，则每次都以该密码为种子加密，否则以随机密钥加密。

如果每次保护时密码相同，则保护后的assets下的文件相同，可以互相替换。



4、so库保护选项，点击选择文件，添加待保护的so库。

隐藏符号表选项：勾选此选项后，可以隐藏so库的导出函数（此选项仅适用于so库列表文件全选）。

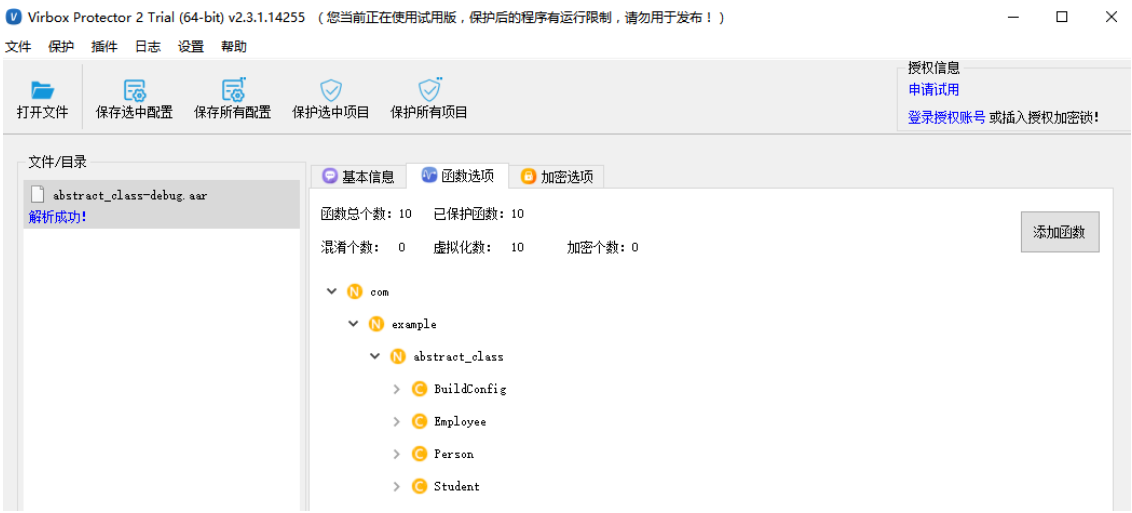
【注】此方式选择的so库只有压缩功能，若有函数保护等需求，需要单独对so库进行保护。

AAR保护

功能：对 JAR 中的方法进行虚拟化等技术保护，保护后的代码无法被还原，防止逆向分析。

操作流程

1. 将aar包直接拖入到工具中；
2. 函数选项处，选择函数为虚拟化；



3. 点击保护项目，即可对aar包进行保护，保护后，程序正常使用即可。

命令行保护

1. 使用Virbox Protector界面工具生成配置文件；
2. 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息；
3. 命令如下：

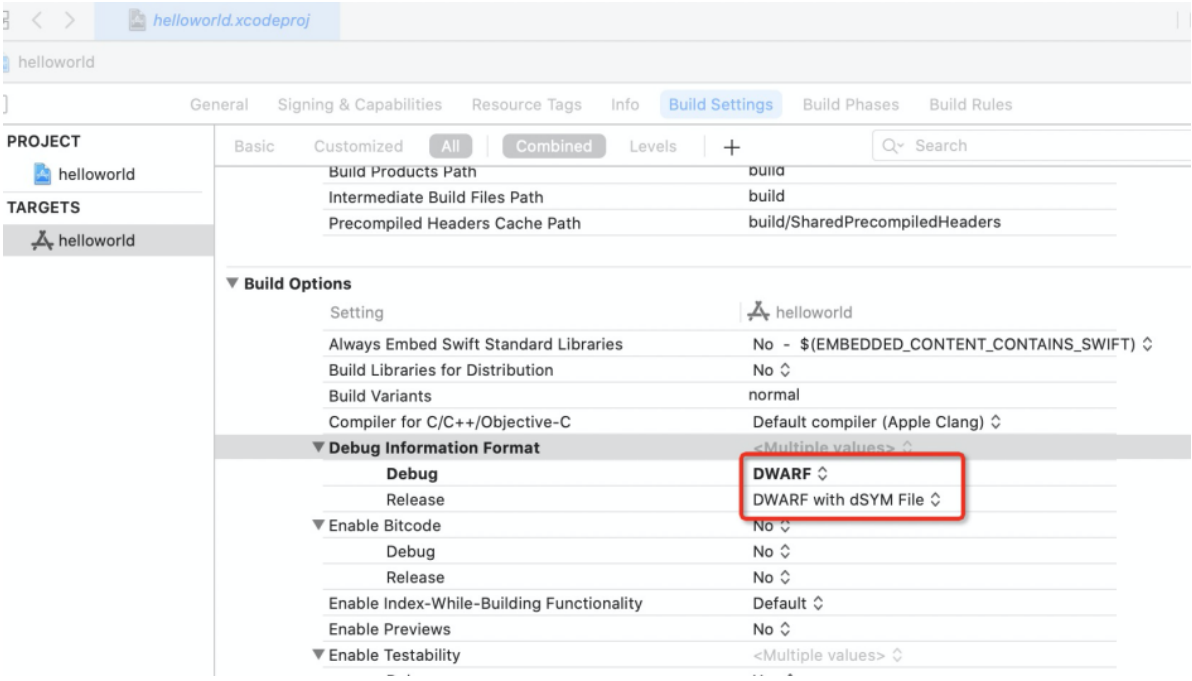
```
virboxprotector_con  <需要被保护的so/apk/aar/aab> -o <输出文件的so/apk/aar/aab>
```

iOS程序保护

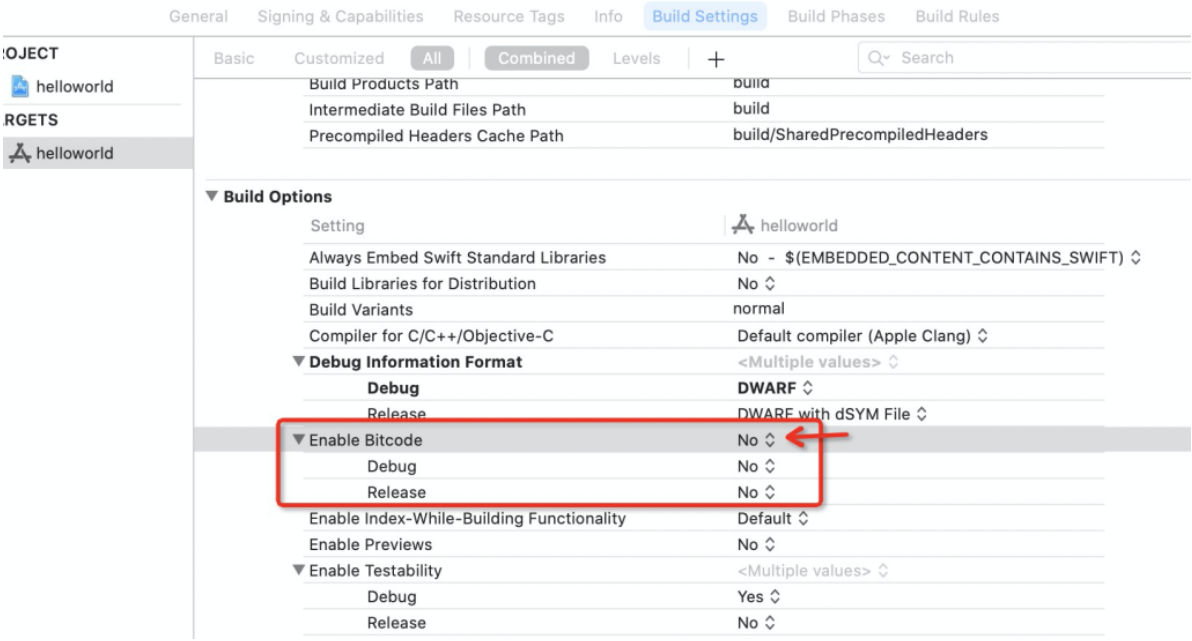
界面使用流程

编译xcarchive包

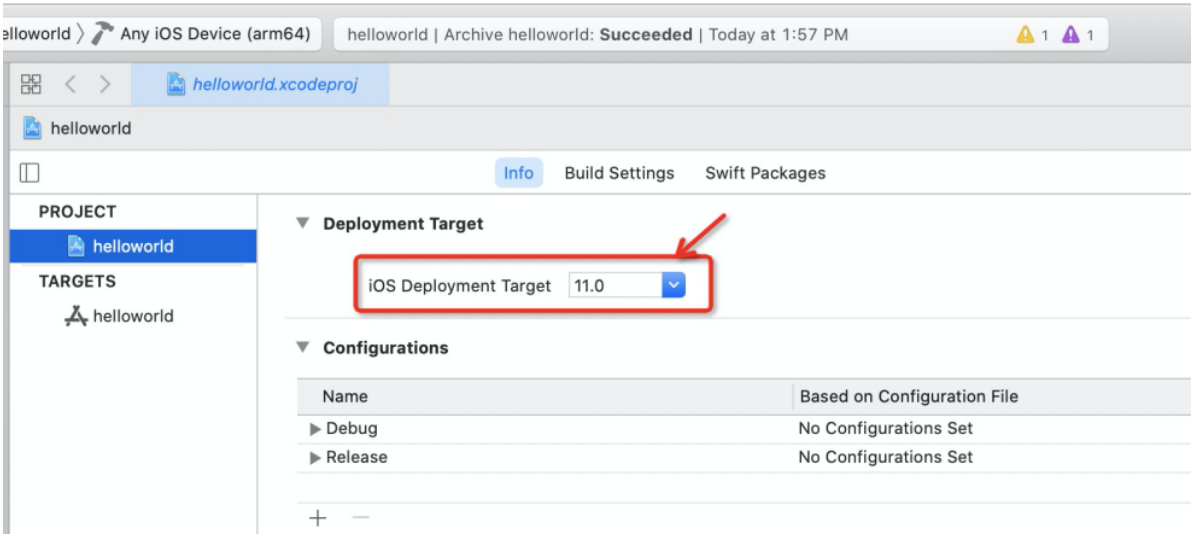
1、Xcode->TARGETS->Build Setting->Build Options->Debug Information Format选择DWARF with dSYM File选项，目的是编译成xcarchive包内带有dSYM文件，加壳工具解析app时可以解析出函数名称，否则，函数将会只会显示地址；



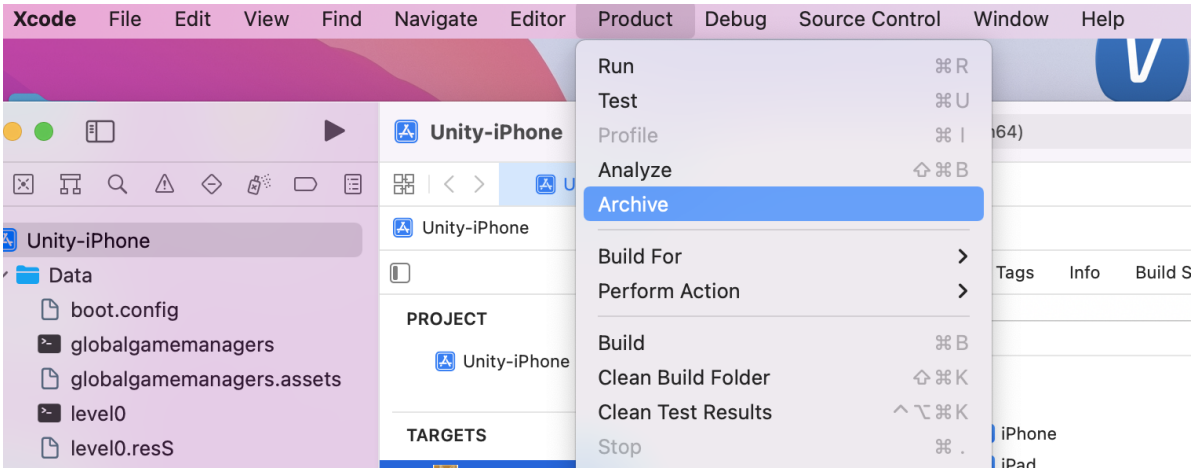
2、加壳工具暂不支持bitcode，编译时关闭bitcode的编译选项，Xcode->TARGETS->Build Setting->Build Options->Enable Bitcode->no；



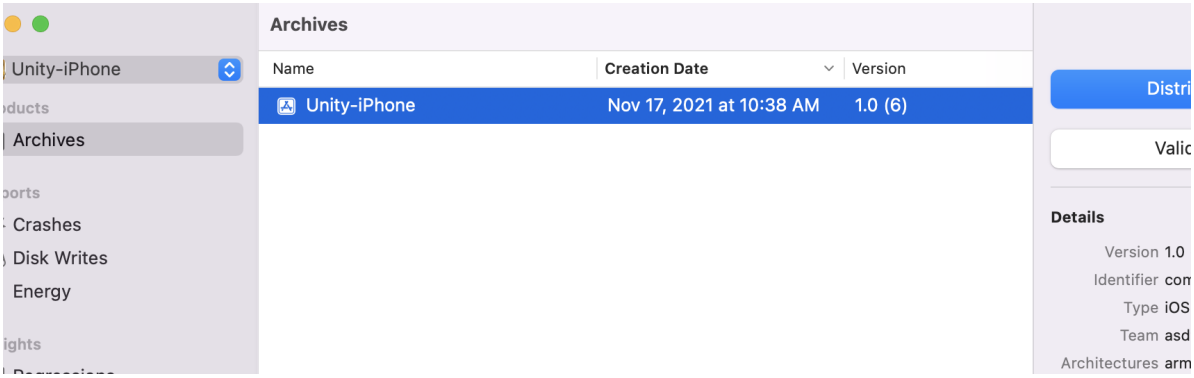
3、加壳暂不支持FAT的格式，编译时请勿开启此模式,方法：ios部署目标选择11.0及以上的版本即可；



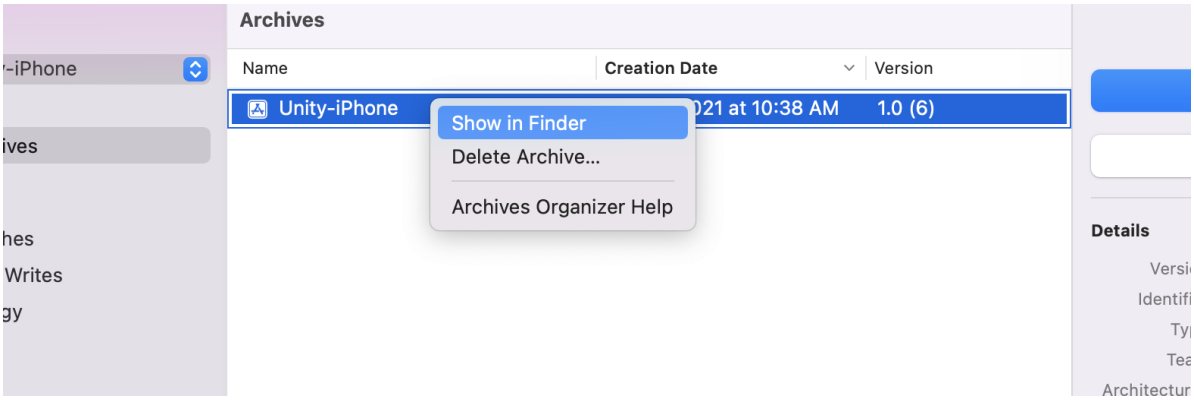
4、以上选项配置完后，选择Xcode->Prodcut->Archive进行编译程序；



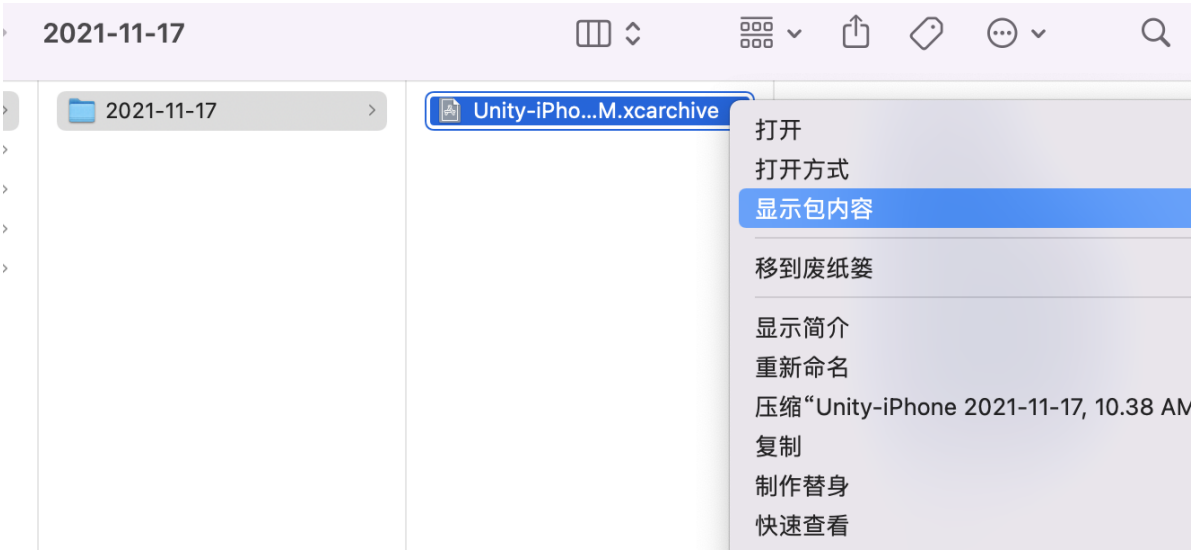
5、编译成功后进入到Archives页面；



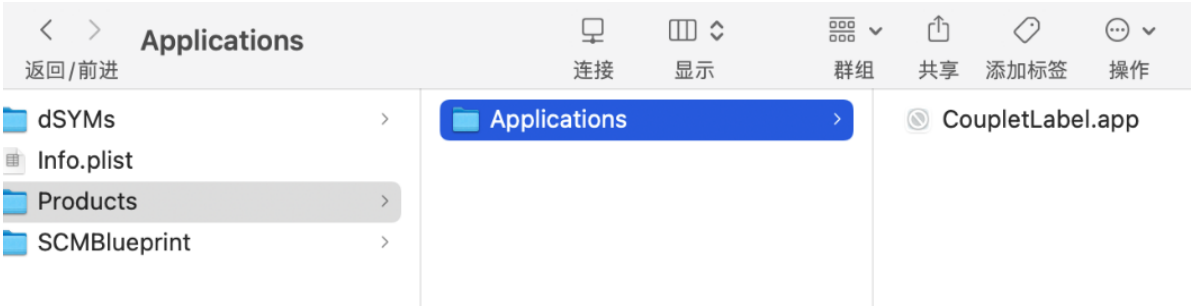
6、选中编译好的程序，右键在Finder里打开该程序；



7、找到编译好的xcarchive，右键显示包内容；



8、进入包内容后，在Products\Applications目录下为待保护的app程序。

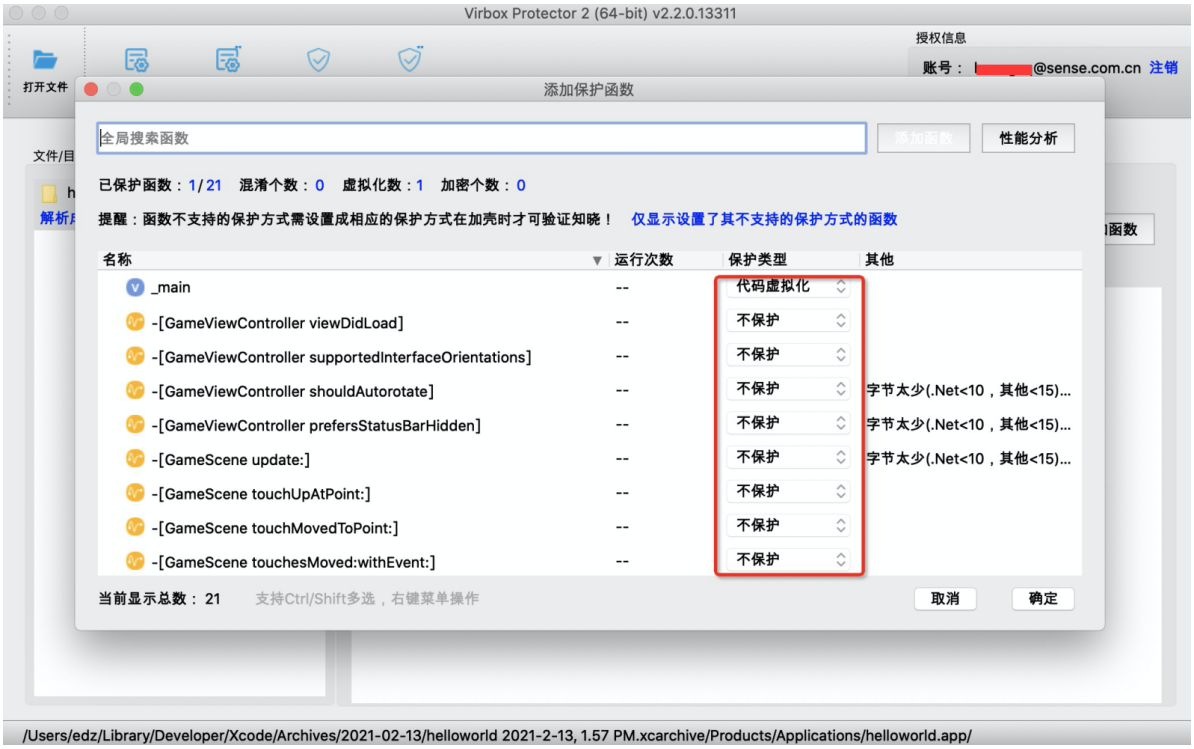


VBP加壳

1、将app程序直接拖入到加壳工具界面；

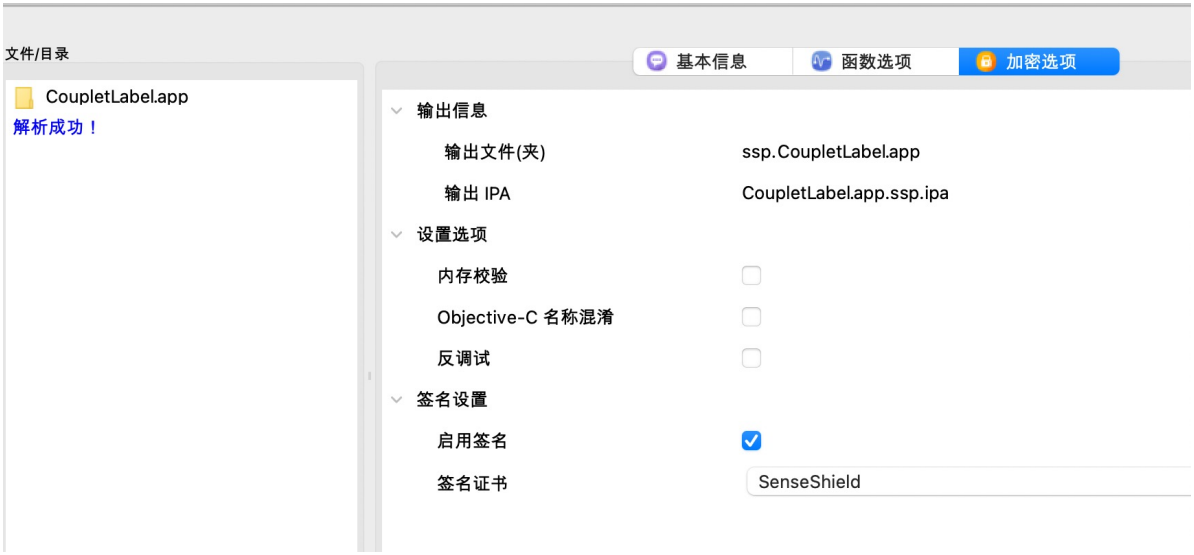
2、单机按钮添加函数可以选择要保护的函数；

3、可以选择想要保护的函数（代码混淆和代码虚拟化）；



4、设置选项功能

- 内存校验：防止反编译工具篡改应用程序；
- Objecttive-C 名称混淆：保护类名，若用class-dump工具dump出.h文件名称是乱码；
- 反调试：可以检测调试器，防止动态调试。



5、设置输出文件名，若勾选启用前面，则程序保护后默认生成IPA文件；

6、选择完成后，单机保护所选项目按钮，等待完成即可。

命令行保护

1. 使用Virbox Protector界面工具生成配置文件(若无配置文件，则命令行加壳后的app默认不签名)；
2. 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息；
3. 命令如下：

```
virboxprotector_con  <需要被保护的app> -o <输出文件的app>
```

Java 程序保护

1. Java 源代码编译后生成的 class 文件由于包含类名、方法名、变量等信息，很容易被反编译，且反编译后几乎可以达到与源代码一致的效果。
2. Virbox Protector 支持对 Java 的 JAR 包、WAR 包进行保护。通过加密 Java 中每个方法的字节码防止反编译，操作简单，运行环境易部署，支持当前主流的 Windows、Linux、ARM Linux 平台。

java虚拟化

java虚拟化将 JVM 字节码转换为自定义的虚拟机指令，运行时跳转至 Native 虚拟机中执行，安全强度高，无法被任何已知工具还原出原始 Java 代码。

【注意】

1. 暂不支持内嵌的 jar;

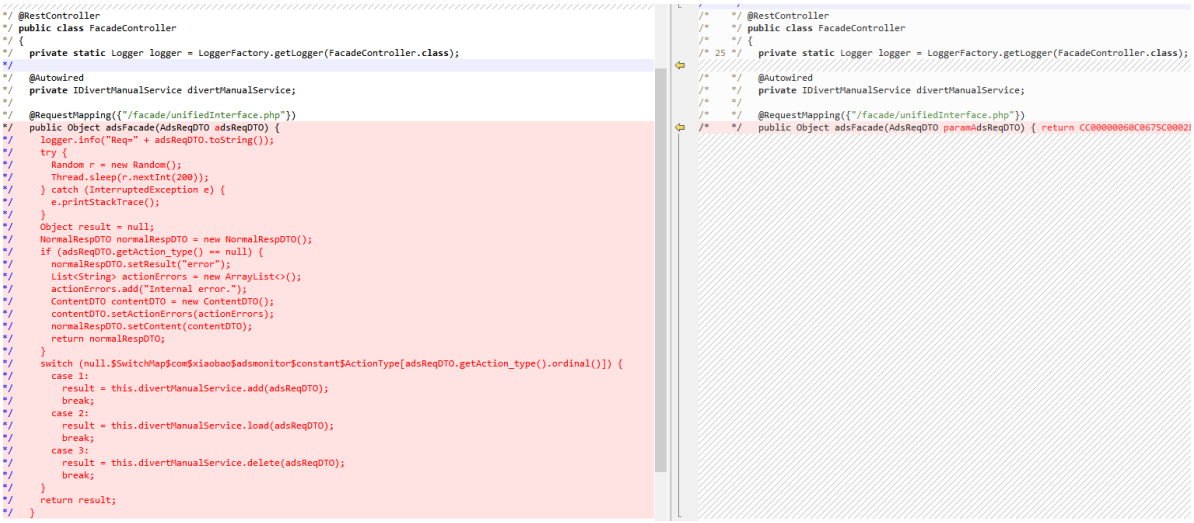
2. 使用了 springframework 的 jar 包，不支持保护 org/springframework/boot/loader 下的 class，仅支持核心 class 文件(如 BOOT-INF/classes 下的 class);
3. 不支持的函数类型包括构造方法、析构方法、使用了反射的方法。

操作流程

1. 将jar包直接拖入到工具中；
2. 函数选项处，选择函数为虚拟化；

3. 点击保护项目，即可对jar包进行保护，保护后，程序正常使用即可。

保护效果



命令行保护

1. 使用Virbox Protector界面工具生成配置文件；
2. 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息；
3. 命令如下：

```
virboxprotector_con  <需要被保护的jar> -o <输出文件的jar>
```

java标签保护

JAVA程序支持代码虚拟化函数保护方式，代码中设置VBVirtualize注解，并在函数上引用，程序编译成功后，将编译好的程序拖入到加壳工具界面，界面会显示代码中设置的函数保护方式：

1、新建VBVirtualize.java，内容是：

```
package virbox;

public @interface VBVirtualize
{
}
```

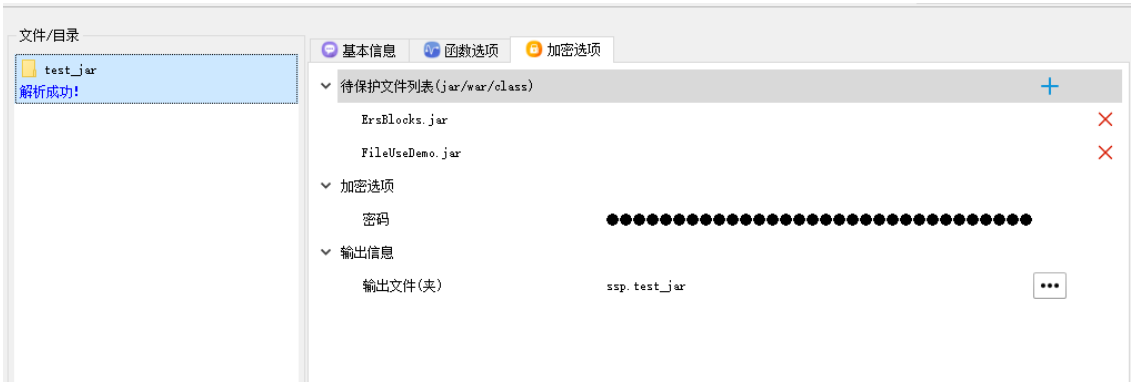
2：调用方式：

```
import virbox.VBVirtualize;

@VBVirtualize //可添加到类上面，所有的方法都会默认保护
public class Main {
    public static void main(String[] args) {
        System.out.println("hello");
        test_vir();
    }

    @VBVirtualize //可添加到方法上面，只保护该方法
    public static void test_vir()
    {
        System.out.println("test_vir");
    }

}
```

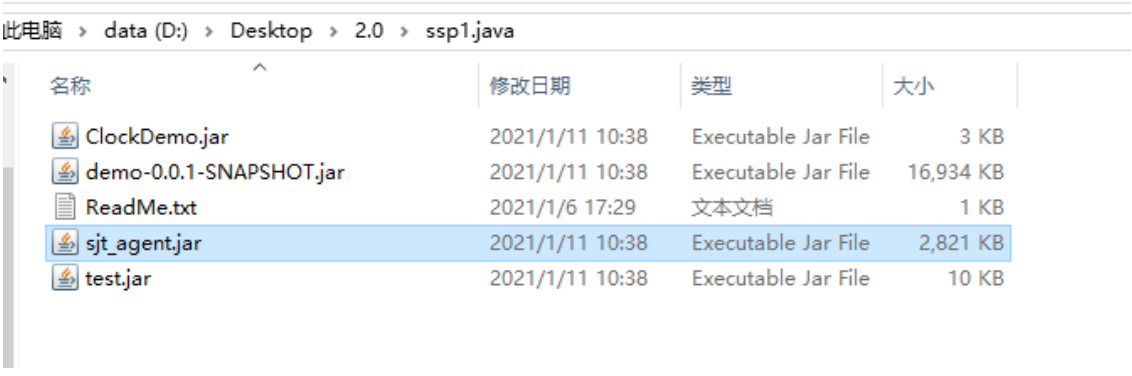
如何对 jar 包保护

操作流程

1. 将包含jar包的目录拖入工具中，直接进行保护。



2. 保护成功后会重新生成一个目录，里面包含加密后的jar包和 sjt 插件。



部署方法

Windows系统

- 直接运行加壳后的程序。

1）若sjt库和jar包在同一目录，进入到jar包的当前目录下，直接执行

> 命令：java -javaagent:sjt_agent.jar-jar ***.jar

2）若sjt库和jar包不在同一目录，需要指定文件的全目录。

> 命令：java -javaagent:C:\Users\test\Desktop\sjt\sjt_agent.jar -jar ***.jar

Linux系统

- 直接运行加壳后的程序。

1）若sjt库和jar包在同一目录，进入到jar包的当前目录下，直接执行。

> 命令：java -javaagent:sjt_agent.jar -jar ***.jar

2）若sjt库和jar包不在同一目录，需要指定文件的全目录。

> 命令：java -javaagent:/home/sense/Desktop/sjt_so/sjt_agent.jar -jar ***.jar

macOS系统

- 直接运行加壳后的程序

1）若sjt库和jar包在同一目录，进入到jar包的当前目录下，直接执行。

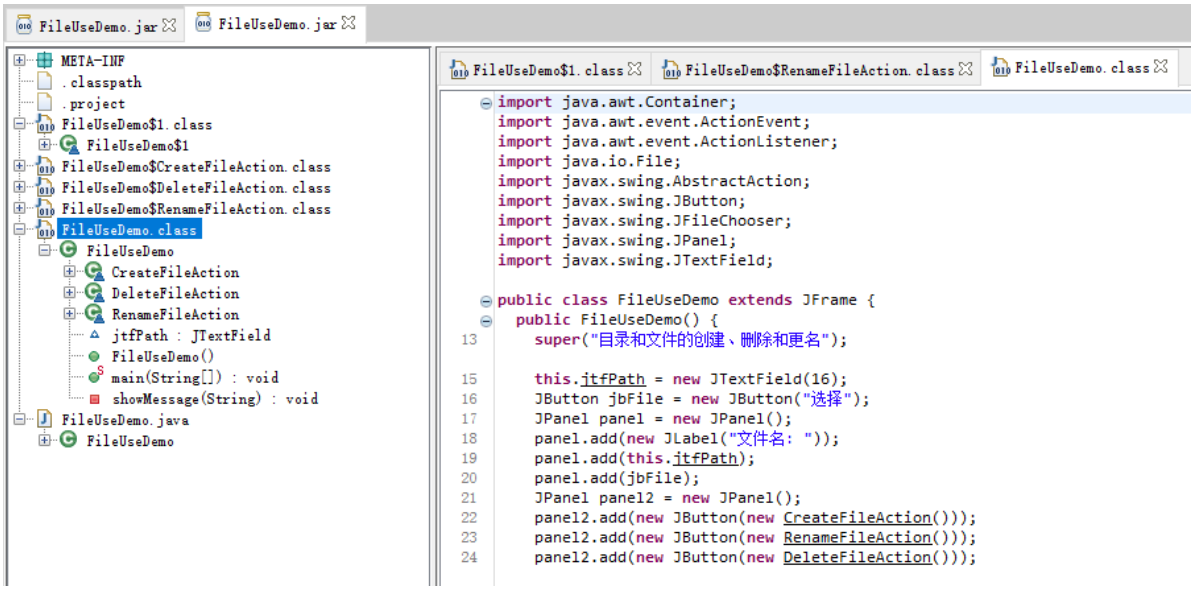
> 命令：java -javaagent:sjt_agent.jar -jar ***.jar

2）若sjt库和jar包不在同一目录，需要指定文件的全目录。

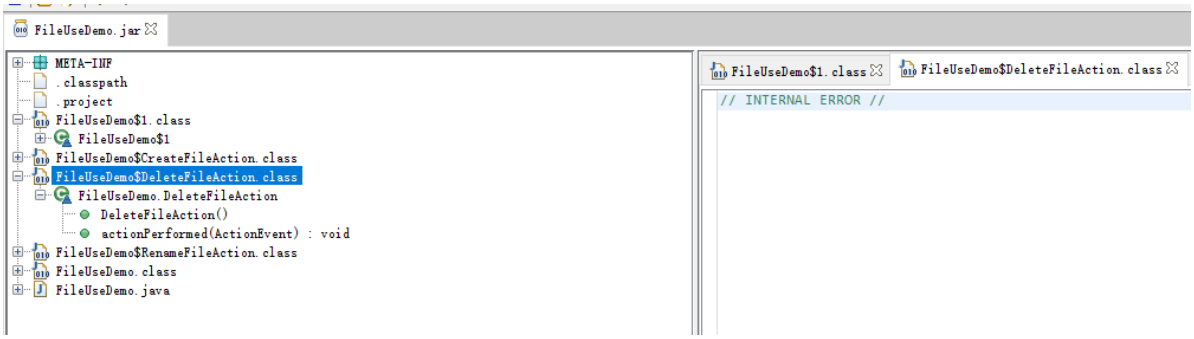
> 命令：java -javaagent:/Users/sense/sjt/sjt_agent.jar -jar ***.jar

jar包保护效果

使用 jd-gui 对 JAR 包反编译效果：



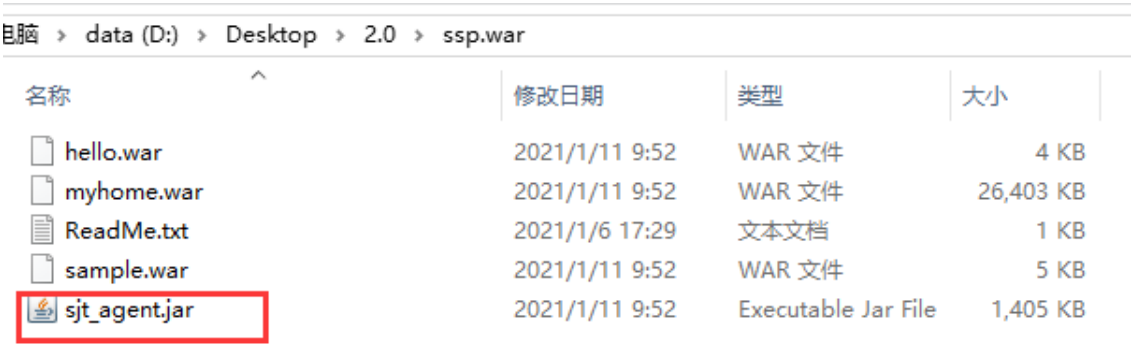
使用 Virbox Protector 保护后的 JAR 包反编译效果：



如何对 war 包保护

操作流程

1. 将war包目录拖入工具中，直接进行保护。
2. 保护成功后会生成文件和sjt插件。



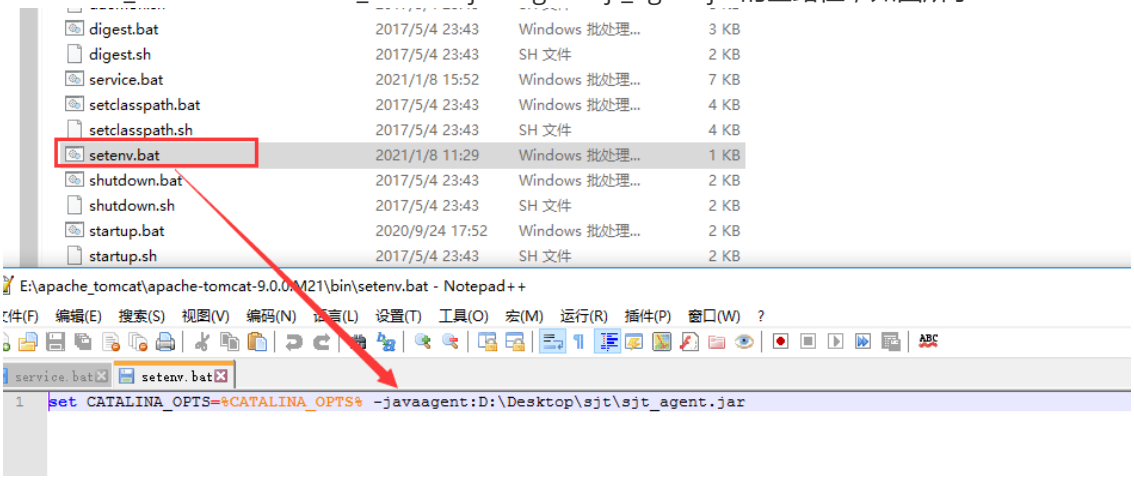
部署方法

Windows系统

以下配置方法不能互相兼容，需选其一：

- tomcat目录下设置setenv.bat

  1）在tomcat\bin目录下新建setenv.bat，文件中设置环境变量比如，set CATALINA_OPTS=%CATALINA_OPTS% -javaagent:sjt_agent.jar的全路径，如图所示：



  2）将保护后的war包放入到 `.\apache-tomcat\webapps` 文件夹中，直接启动tomcat服务即可运行。

- 若启动tomcat时在服务中启动

  1）首选需要将tomcat服务给卸载，控制台命令service.bat uninstall；

  2）在service.bat里的JvmOptions参数里加上sjt库，如图所示：

```
"%EXECUTABLE%" //IS//%SERVICE_NAME% ^
    --Description "Apache Tomcat 9.0.0.M21 Server - http://tomcat.apache.org/" ^
    --DisplayName "%DISPLAYNAME%" ^
    --Install "%EXECUTABLE%" ^
    --LogPath "%CATALINA_BASE%\logs" ^
    --StdOutput auto ^
    --StdError auto ^
    --Classpath "%CLASSPATH%" ^
    --Jvm "%JVM%" ^
    --StartMode jvm ^
    --StopMode jvm ^
    --StartPath "%CATALINA_HOME%" ^
    --StopPath "%CATALINA_HOME%" ^
    --StartClass org.apache.catalina.startup.Bootstrap ^
    --StopClass org.apache.catalina.startup.Bootstrap ^
    --StartParams start ^
    --StopParams stop ^
    --JvmOptions "-Dcatalina.home=%CATALINA_HOME%;-Dcatalina.base=%CATALINA_BASE%;-Djava.io.tmpdir=%CATALINA_BASE%
    \temp;-Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager;-Djava.util.logging.config.file=%CATALINA_BASE%
    \conf\logging.properties;-javaagent:D:\Desktop\sjt\sjt_agent.jar;%JvmArgs%" ^
    --Startup "%SERVICE_STARTUP_MODE%" ^
    --JvmMs "%JvmMs%" ^
    --JvmMx "%JvmMx%"
if not errorlevel 1 goto installed
echo Failed installing '%SERVICE_NAME%' service
goto end
:installed
echo The service '%SERVICE_NAME%' has been installed.

:end
cd "%CURRENT_DIR%"
```

  3）然后在控制台命令行里service.bat install；

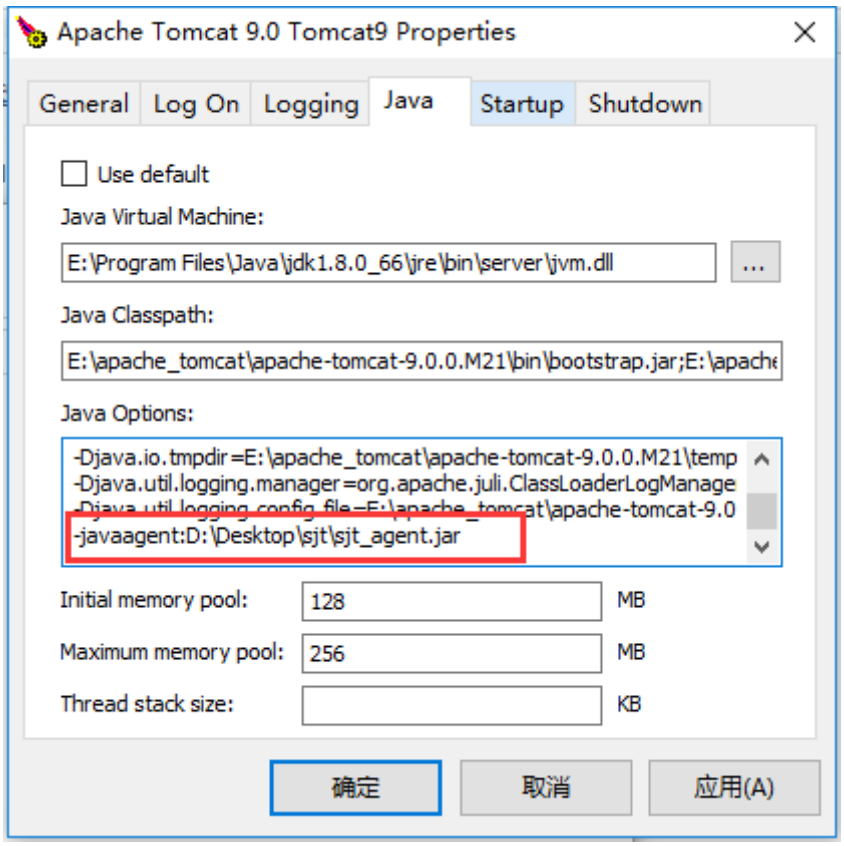
  4）然后服务里启动tomcat ；

  5）将保护后的war包放入到 `.\apache-tomcat\webapps` 文件夹中，直接启动tomcat服务即可运行。

- 若是直接使用tomcat9.exe启动服务

  1）首先启动tomcat9w.exe；

  2）在Java Options操作列表添加sjt库，如图所示：



3）然后运行tomcat9.exe启动tomcat服务，将保护后的war包放入到 `.\apache-tomcat\webapps` 文件夹中，直接启动tomcat服务即可运行。

Linux系统

- tomcat目录下设置setenv.sh

  1）在tomcat\bin目录下新建setenv.sh，文件中设置环境变量比如CATALINA_OPTS="$CATALINA_OPTS -javaagent:sjt_agent.jar"的全路径，如图所示：

```
CATALINA_PID="$CATALINA_BASE/tomcat.pid"
export CATALINA_OPTS="$CATALINA_OPTS -javaagent:/home/sense/Desktop/sjt_so/sjt_a
gent.jar"

~
~
```

  2）启动tomcat服务，可以查看到设置的CATALINA_OPTS参数。

```
root@sense:/usr/local/apache-tomcat-8.5.58/bin# ./startup.sh
Using CATALINA_BASE:   /usr/local/apache-tomcat-8.5.58
Using CATALINA_HOME:   /usr/local/apache-tomcat-8.5.58
Using CATALINA_TMPDIR: /usr/local/apache-tomcat-8.5.58/temp
Using JRE_HOME:        /usr/local/java/jdk1.8.0_251/jre
Using CLASSPATH:       /usr/local/apache-tomcat-8.5.58/bin/bootstrap.jar:/usr/lo
cal/apache-tomcat-8.5.58/bin/tomcat-juli.jar
Using CATALINA_OPTS:    -javaagent:/home/sense/Desktop/sjt_so/sjt_agent.jar
Using CATALINA_PID:    /usr/local/apache-tomcat-8.5.58/tomcat.pid
Existing PID file found during start.
Removing/clearing stale PID file.
Tomcat started.
```

  3）将保护后的war包放入到 `.\apache-tomcat\webapps` 文件夹中，tomcat服务正常解析war包，网页即可运行。

注：若配置系统环境变量后，即使指定sjt库的位置，java运行也会先走系统环境变量里的配置。

macOS系统

1、在tomcat/bin目录下新建setenv.sh，文件中设置环境变量比如CATALINA_OPTS="$CATALINA_OPTS -javaagent:/Users/sense/sjt/sjt_agent.jar"。

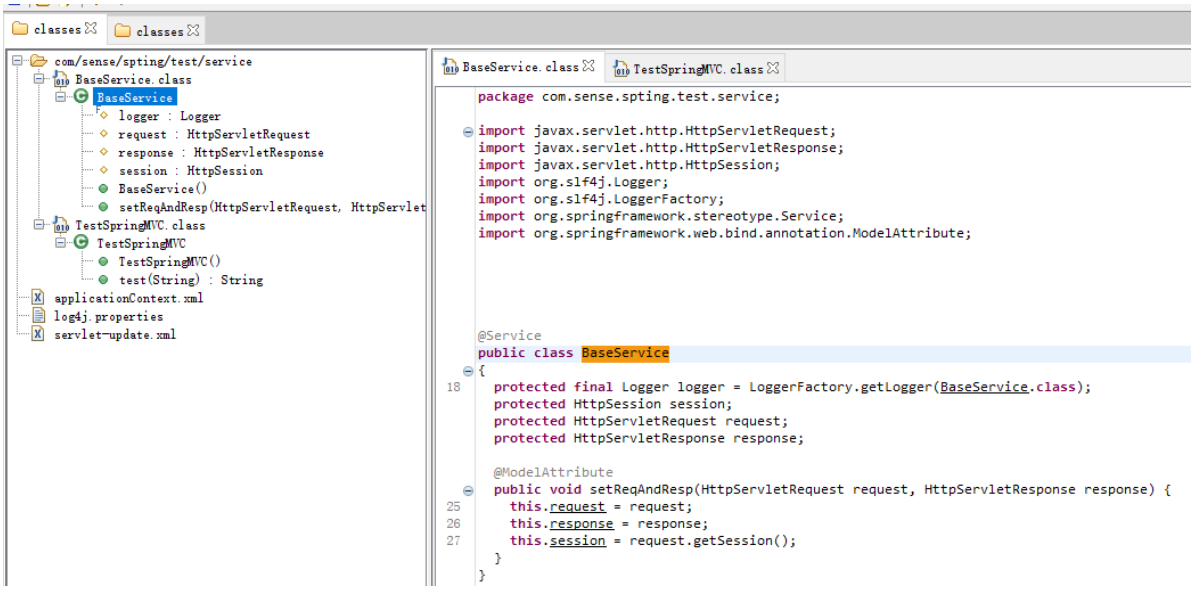
2、启动tomcat服务，可以查看到设置的CATALINA_OPTS参数。

3、将保护后的war包放入到 `.\apache-tomcat\webapps` 文件夹中，tomcat服务正常解析war包，网页即可运行。

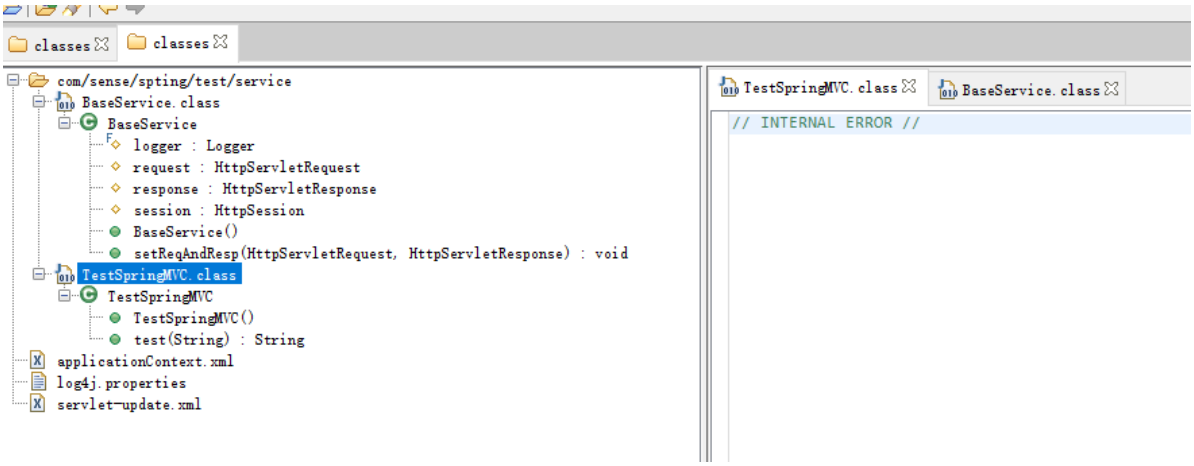
war包保护效果

保护方式：目前只对war包里的class文件进行加密保护。

- 使用 jd-gui 对 war包中class文件反编译效果：



- 使用 Virbox Protector 保护后的 war 包中class文件反编译效果：



使用命令行工具保护

Java作为一个特殊的程序，和普通程序的保护方式不同，针对Windows、Linux和macOS平台的Java，需要对Java整个目录进行保护。

1. 使用Virbox Protector界面工具生成配置文件；
2. 打开终端窗口，进入到“virboxprotector_con”所在的路径，直接输入“virboxprotector_con”运行可查看帮助信息；
3. 针对不同平台的程序，独立壳对其许可的限制不同，需要联系**深思销售**获取许可；

```
virboxprotector_con -java <需要被保护的程序目录> --password <password> -o <输出文件的目录>
```

Python/PHP 等脚本语言保护

请参考同目录下的《DS.pdf》文档。

常见问题

加密后的软件被杀毒软件拦截

1. 问题描述：使用加壳工具对开发者软件进行加壳，然而加密后的程序被 360 等杀毒软件认为是病毒软件。
2. 解决方法：提交 360 认证

   1）软件中可执行文件（不包含驱动程序 .sys）及打包后的可执行文件使用沃通（Wosign）签名（使用沃通代码签名工具并且购买代码签名证书进行签名，沃通签名相关问题请咨询沃通官方网站客服人员，沃通签名在 360 认证过程有很大帮助）。

   2）注册并登录 360 开放平台：http://open.soft.360.cn/

   3）在“我的软件”页面中选择提交我的软件，如下图：

   4）选择“仅安全检测”，并填写软件名称，软件版本（主要为了后续记录查看），提交方式可以选择本地上传安装包，然后提交软件。

   5）同时可以在“软件列表”中查看已经提交过的待检测软件和“通过检测”的软件。目前软件提交后通过检测的时间大约为1天。

不支持列表

由于程序代码语言、书写规范、平台和架构等影响编译出的程序有部分Virbox Protector目前不支持保护，请参考下表：

类型		不支持的情况列表
其他		不支持二次加壳，无论是第三方还是本程序加壳后的文件，都不能再次进行加壳
		不支持带有自校验检查的程序
文件类型	.NET	暂不支持带有程序集签名(强签名)的程序进行加壳
		.NET加壳不支持第三方运行时库，只支持微软标准运行时库
		C#开发的.NET程序或DLL库使用名称混淆功能默认只保护私有成员变量，其他方法暂时不保护
		暂不支持勾选Ready To Run选项编译的程序
		Linux和macOS上Dotnet Core3 的dll不支持压缩功能和IT加密
		LM许可壳暂不支持对Linux和macOS上Dotnet Core3 的dll进行加壳
	PE	PPT转exe的程序不支持资源保护
		VB6.0语言程序不能加资源保护
		PE程序的导入表功能，要求导入的符号必须都是函数，不能有导入变量，否则运行时程序会崩溃
		如果被保护的程序使用了内存加载方式执行，压缩后无法运行
	ELF	Linux的程序暂不支持附加数据
		LM版不支持static elf程序的保护
		ELF文件不支持map文件分析
		如果默认选项导出了所有符号，可能在运行时会崩溃，建议只导出需要导出的函数
		pyinstall转成的Linux可执行文件的暂不支持压缩
	Unity3d	macOS arm64平台，暂不支持il2cpp和mono格式的unity3d程序
保护选项	代码加密	由解析器通过引用分析得到的函数（函数列表中没有名称的函数），可能存在外部入口而不支持
		函数指令字节过小，不能保护
	混淆/虚拟化/碎片化	对于 ELF 和 Mach-O 格式的程序，如果函数被优化为使用了“野栈”，则不支持保护
		函数指令字节过小，不能保护
		ARM架构程序不支持碎片化

已知问题

1. .NET 加壳不支持第三方运行时库，只支持微软标准运行时库。
2. 在.NET程序中使用类似 GetField("name", bindingAttr) 函数时，加壳后名称混淆可能出现异常，如果 .NET 程序加壳后运行失败，尝试去掉名称混淆。
3. 对函数块进行碎片化代码保护的时候，存在不能成功保护的情况，主要原因是，碎片代码对指令的长度过小，指令可能不可移植，存在跳转等情况。
4. 如果用户的机子上装有杀毒软件 AVAST，可能会出现加壳后的程序无法运行的情况。导致此问题的原因是，当加壳程序运行的时候 AVAST 会杀死加壳程序的进程。
5. 暂不支持带有程序集签名(强签名)的程序进行加壳。