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

产品名称	描述	收费模式	目标平台	备注
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

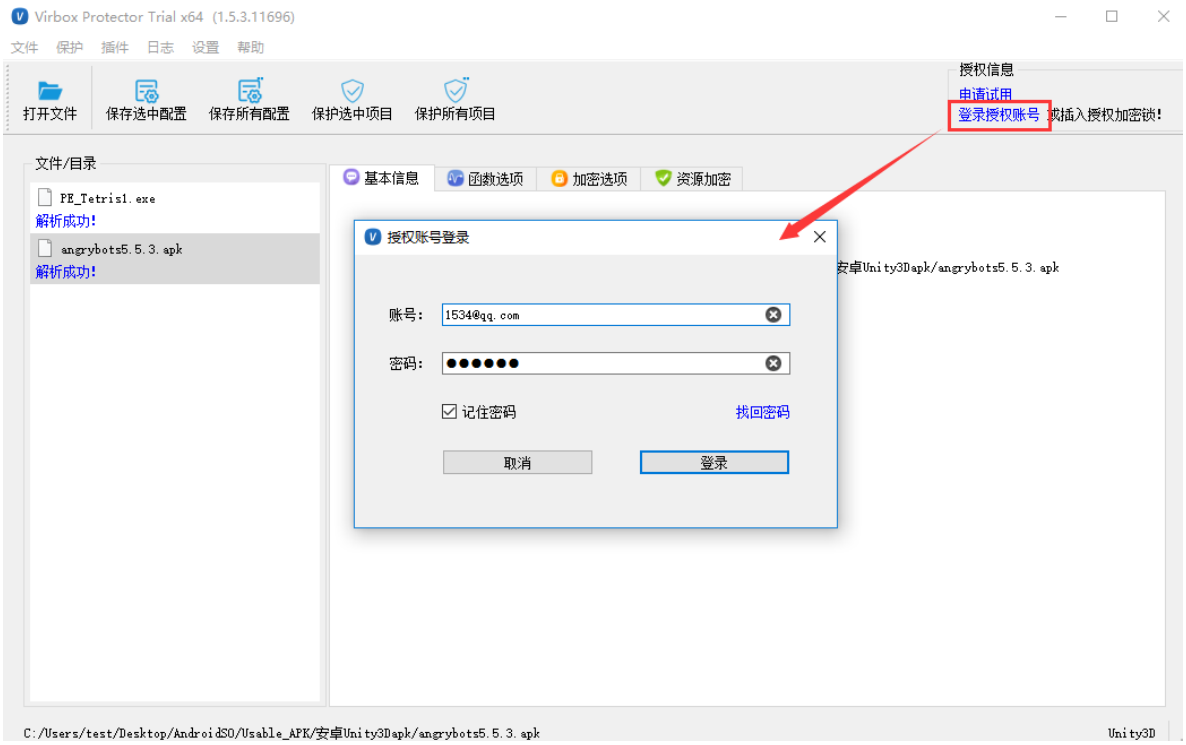
Virbox Protector Trial

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。

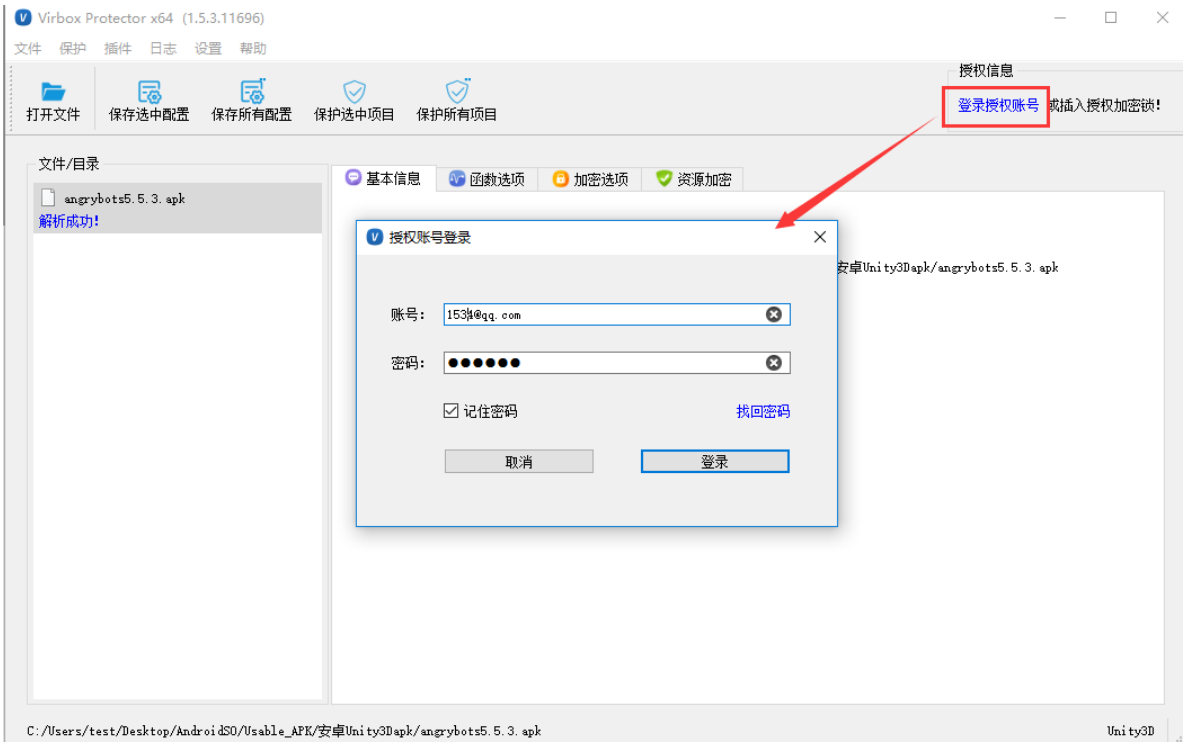
4、若已经申请过拿到授权，点击"登录授权账号"按钮，登录账号后即可对文件进行使用。



Virbox Protector

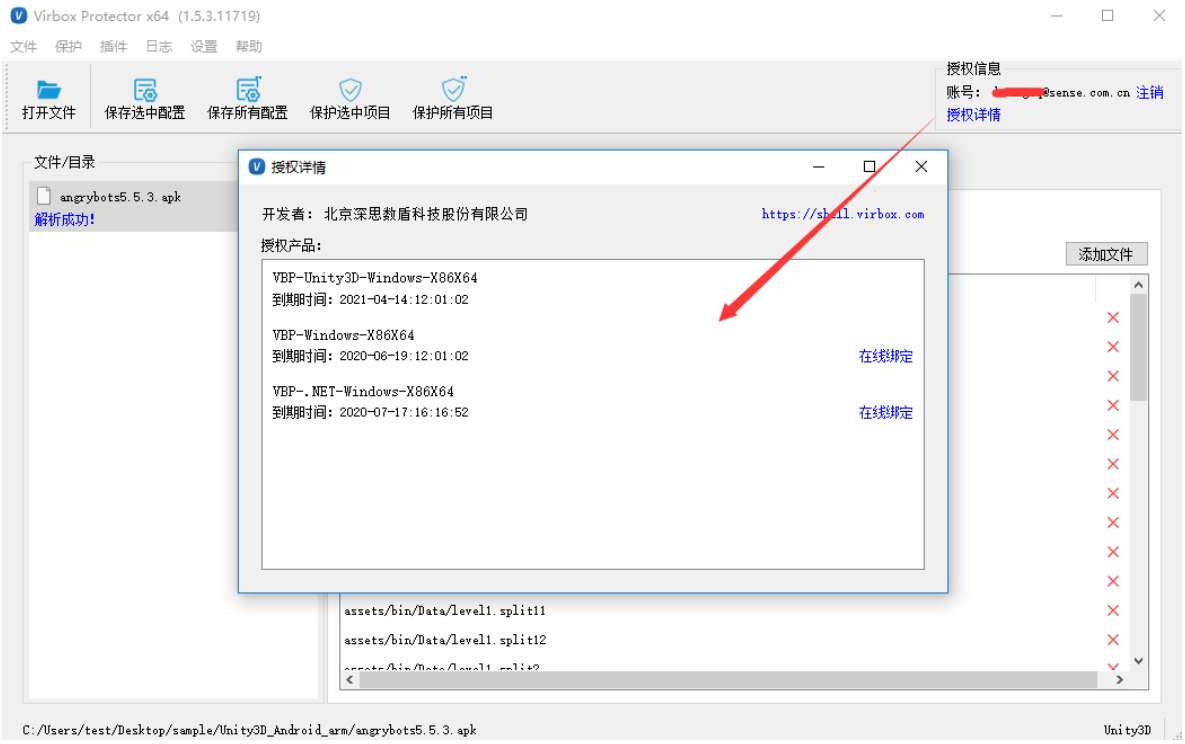
1、需要联系**深思数盾销售人员**获取版本，授权账号为“申请试用”时的账号。

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。

- 授权详情，可以查看到该账号当前的授权信息。
- 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

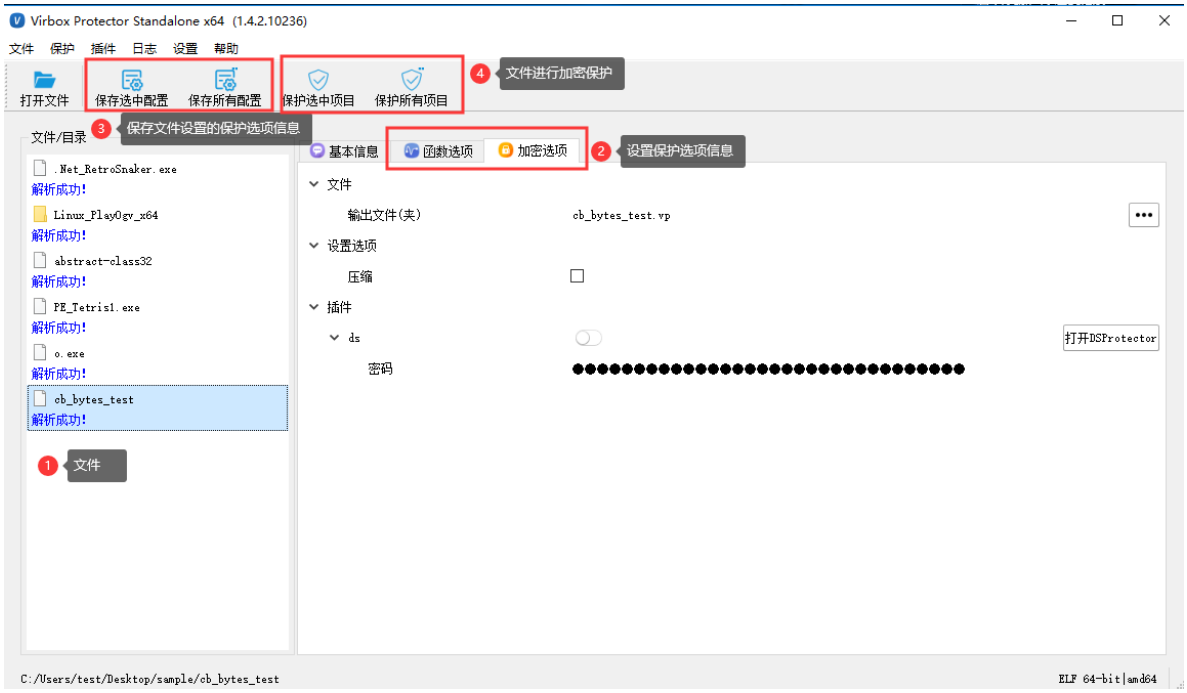
安装目录结构

如下所示:

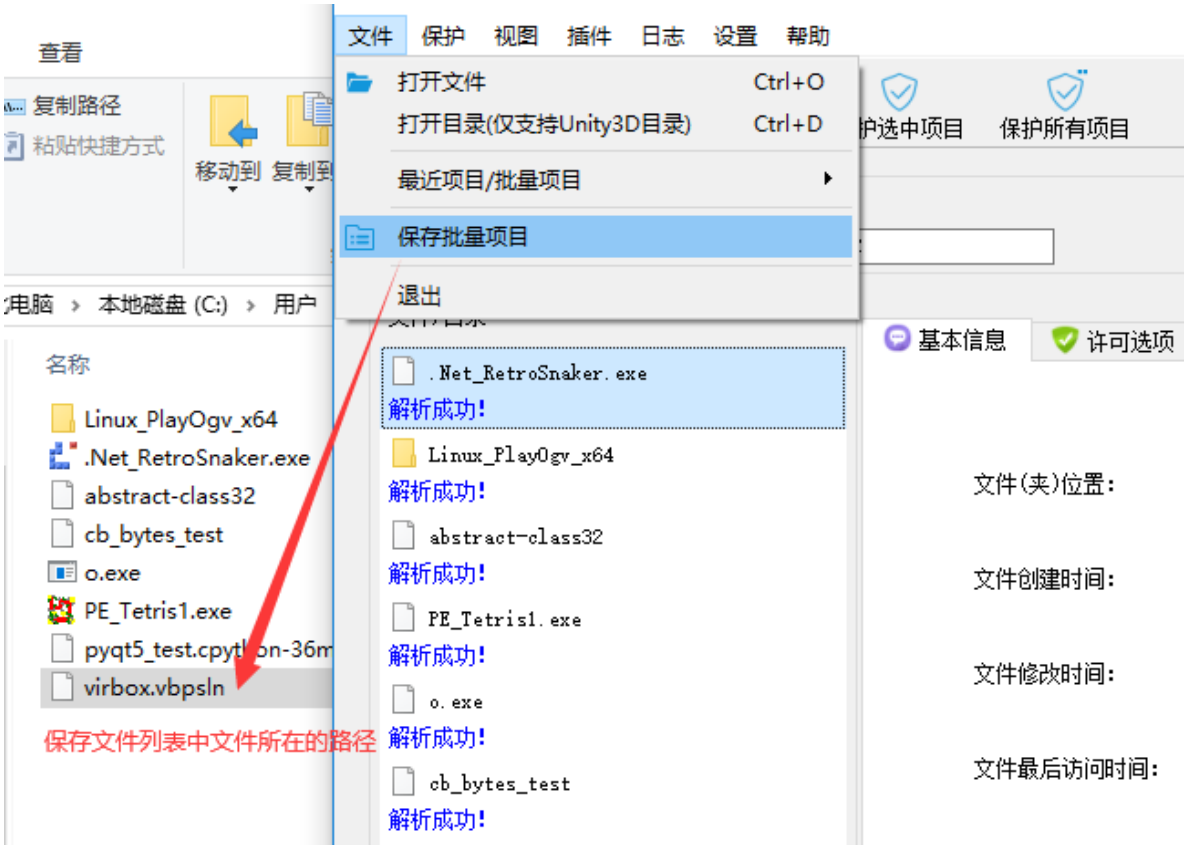
```
├─bin
│  ├─virboxprotector.exe
│  ├─virboxprotector_con.exe
│  └─dsprotector_con.exe
├─example
│  ├─plugin
│  │  └─demo
│  │      └─src
│  └─sdk
├─help
├─plugin
│  ├─anti
│  └─ds
└─sdk
```

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

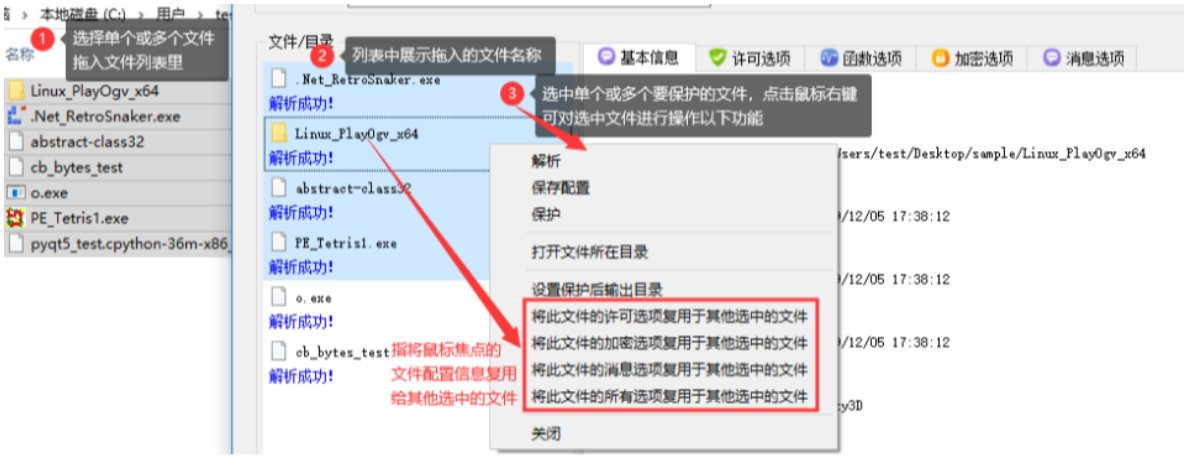
工具栏处显示打开文件、保存选中配置等快捷按钮，在工具栏处右键点击还可以隐藏工具栏。

- 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。
- 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。
- 保护选中项目：指将选中文件进行加密保护。
- 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。

- 将文件拖入到 Virbox Protector 工具界面，工具界面展示文件信息。
- 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。
- 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。
    - 打开文件所在目录：指打开鼠标焦点的文件所在目录。
    - 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

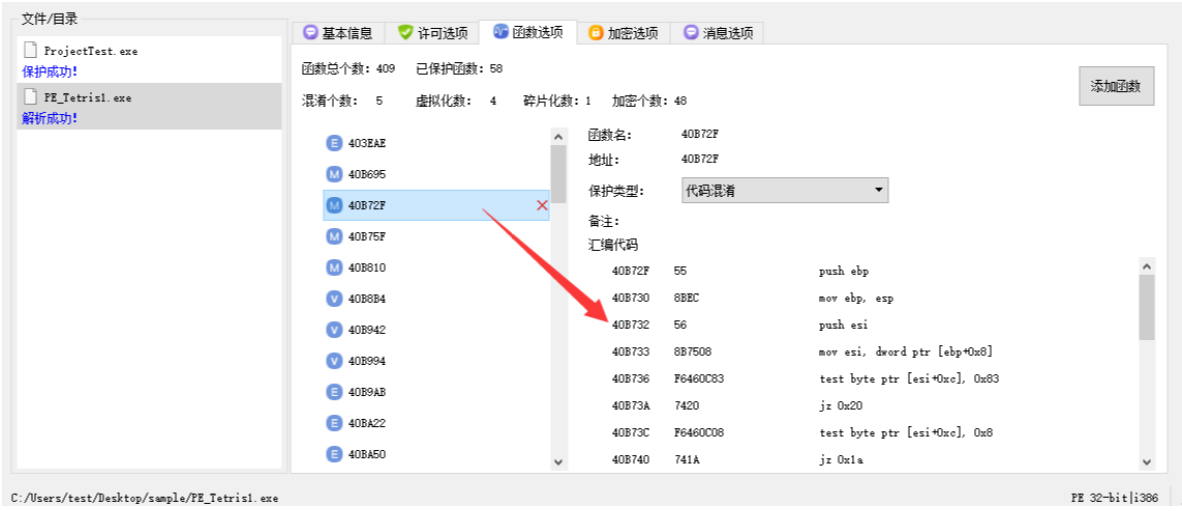
主要显示文件(夹)位置、文件创建时间、文件修改时间、文件最后访问时间和文件类型信息。

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。

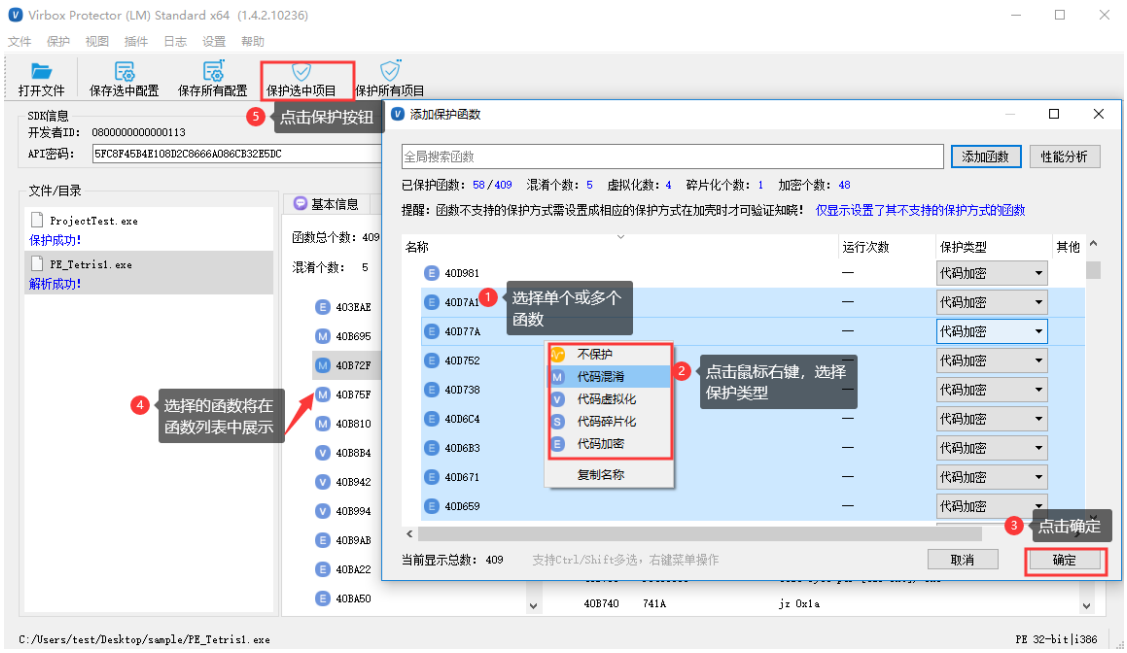
查看函数的详细信息

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。

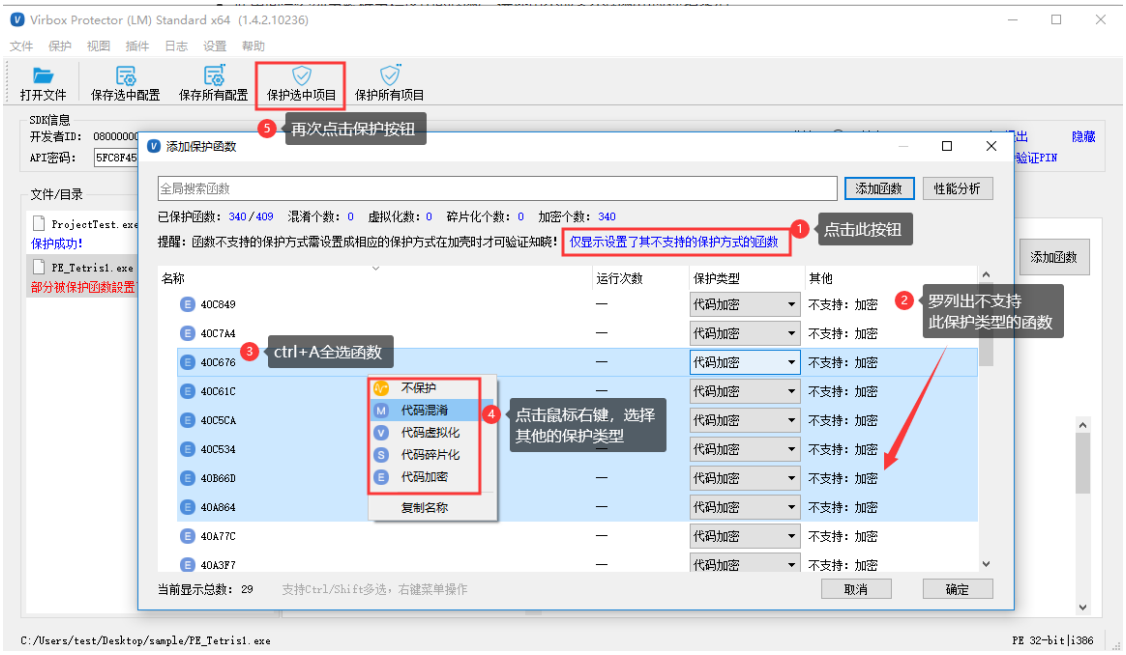


添加函数

- 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。
- 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。
- 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：



- 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

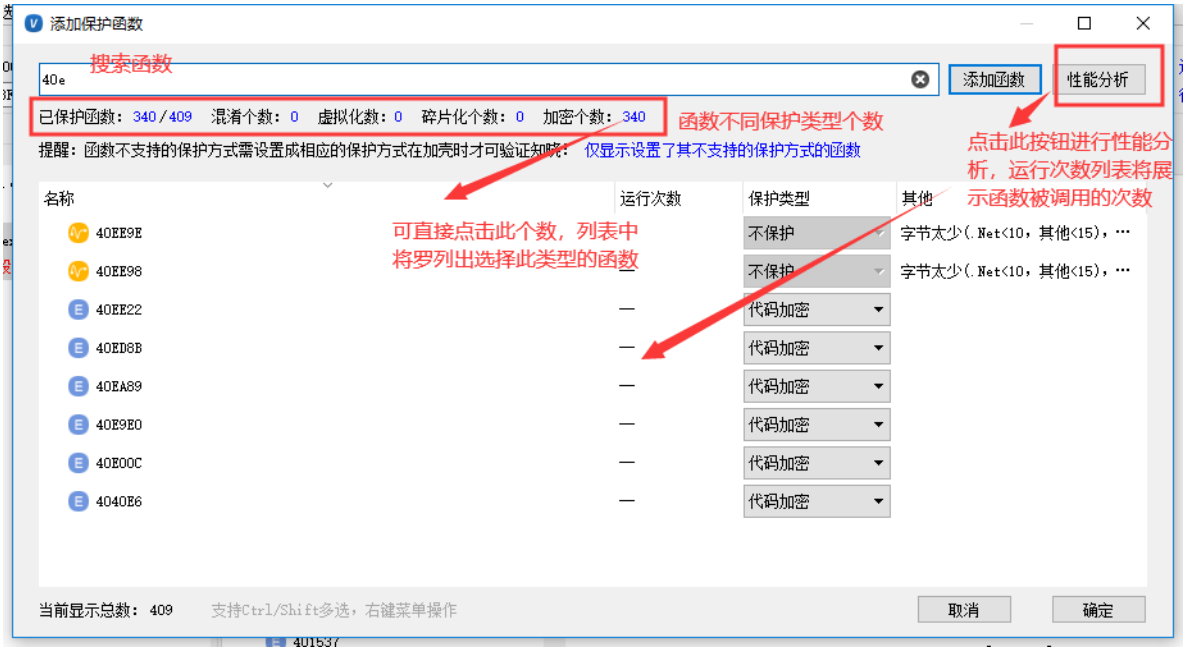
在搜索框中输入关键字，程序会列出所有名称中包含关键字的函数块，支持模糊查询。

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ll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。

反调试插件

支持平台：Windows、Linux、ARM Linux、Android so和Android Unity3D。

Windows

反调试插件包括检测硬件断点、检测内存断点和内存检查，主要防止ollydbg、windbg等工具进行调试。

【注意】代码加密和内存检测为互斥关系，ds和内存检测为互斥关系。

- 检测硬件断点，可以检测程序中是否设置硬件断点，若检测到程序中设置内存访问断点和内存写入断点时，则程序直接终止运行。
- 检测内存断点，可以检测程序中是否设置内存断点，若检测到则程序直接终止运行。
- 内存检查，可以检测到内存是否被修改（比如被调试器附加修改），若程序内存被修改则程序将终止运行。

Linux、ARM Linux、Android so和Android Unity3D

反调试插件包括检测调试器功能，主要防止保护后的程序被反编译工具(如gdb、IDA等)进行调试。

勾选此功能后，使用gdb调试保护后的程序的效果，如图所示：

```
(gdb) r
Starting program: /home/sense/Desktop/0509antitest/asrproxy/asrproxy
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
This application is protected with unregistered version of VirboxProtector. 6 da
ys left

Program received signal SIGABRT, Aborted.
__GI_raise (sig=sig@entry=6) at ../sysdeps/unix/sysv/linux/raise.c:50
50      ../sysdeps/unix/sysv/linux/raise.c: No such file or directory.
(gdb) c
Continuing.

Program terminated with signal SIGABRT, Aborted.
The program no longer exists.
(gdb)
```

勾选此功能后，使用IDA工具调试保护后的程序的效果，如图所示：

```
Output window
F1FFAB1C: thread has started (tid=28720)
Debugger: thread 28720 has exited (code 0)
F1FFAB1C: thread has started (tid=28721)
D3515000: loaded /data/app/com.KLS.LetteClear-Xgwvm2xcylu6QvChWmk7Hg==/lib/arm/libmain.so
D33C7000: loaded /system/framework/oat/arm/gson.odex
D381E3EA: got SIGSEGV signal (Segmentation violation) (exc.code b, tid 28663)
F204124C: got SIGSEGV signal (Segmentation violation) (exc.code b, tid 28663)
D2F00000: loaded /data/app/com.KLS.LetteClear-Xgwvm2xcylu6QvChWmk7Hg==/lib/arm/libmono.so
F1FFDFAA: got SIGABRT signal (Abort) (exc.code 6, tid 28663)
F1FFDFDC: got SIGABRT signal (Abort) (exc.code 6, tid 28663)
```

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。

本地可执行程序保护

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。

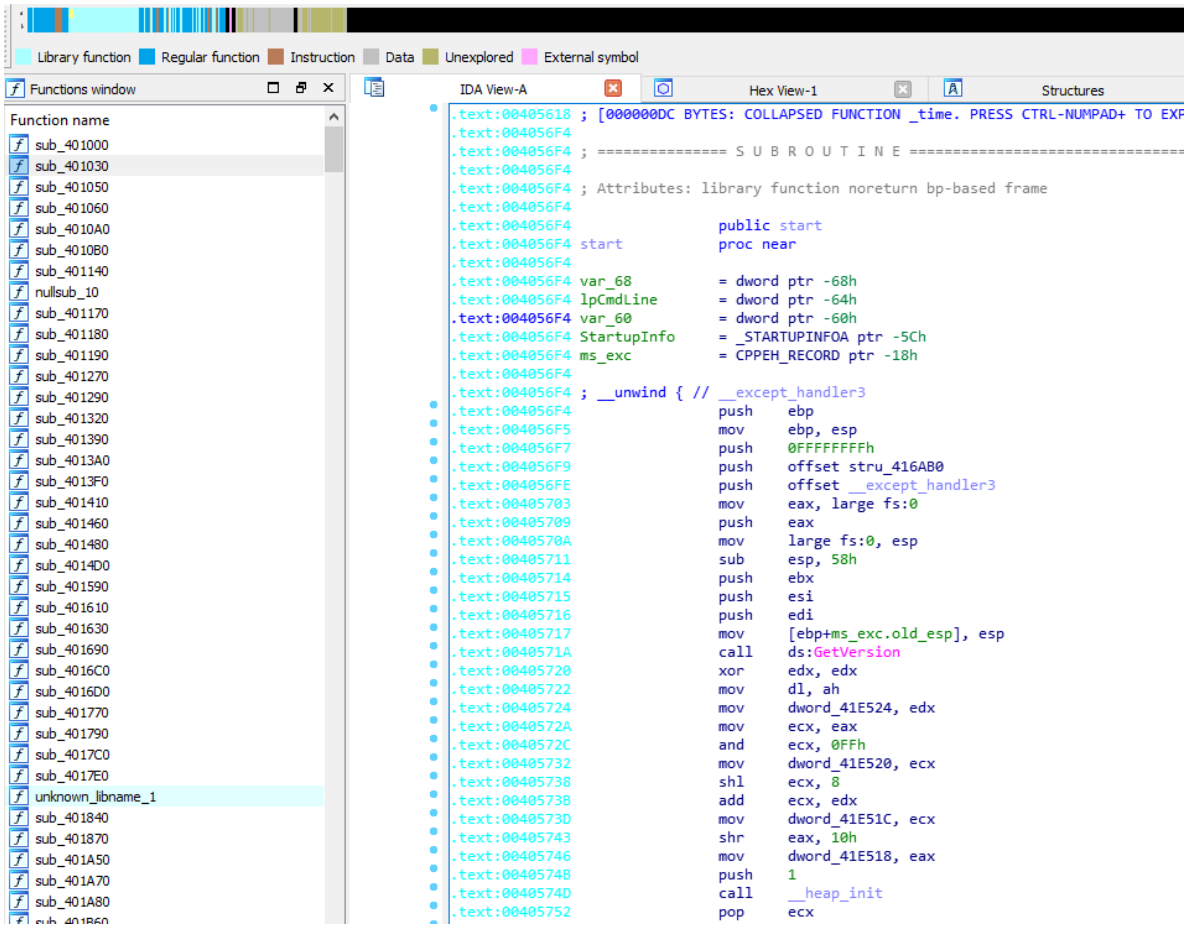
  2、运行效率高，仅在程序被加载时轻微的性能损失。

- **缺点**

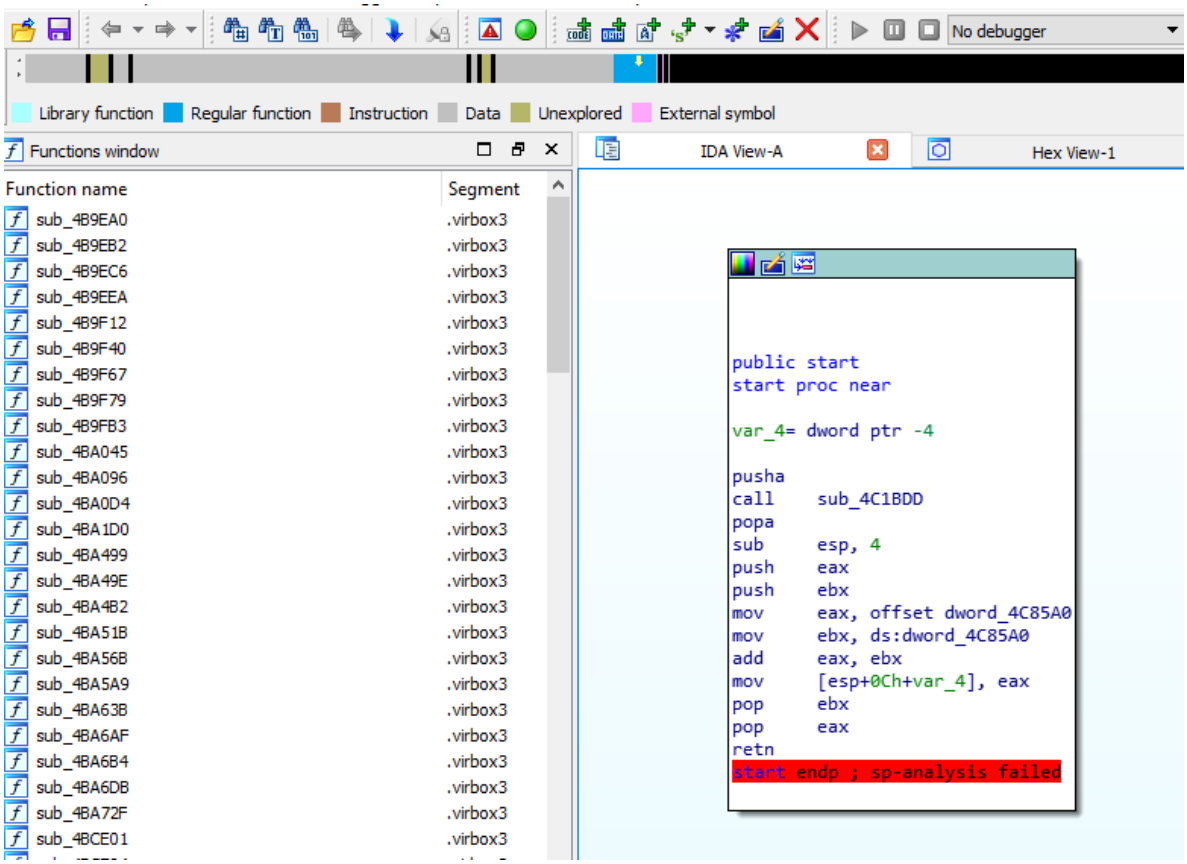
  1、壳代码执行完毕后，代码段与数据段会还原，可以被 Dump。

保护效果图

- 保护前，如图所示：



- 保护后，如图所示：



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。

保护效果图

- x86架构程序保护前，如图所示：

```
.text:004144B5 ; __unwind { // loc_414CD8
.text:004144B5                 mov     eax, offset loc_414CD8
.text:004144BA                 call    __EH_prolog
.text:004144BF                 push    ecx
.text:004144C0                 mov     [ebp+var_10], ecx
.text:004144C3                 mov     dword ptr [ecx], offset off_416508
.text:004144C9 ;   try {
.text:004144C9                 and     [ebp+var_4], 0
.text:004144CD                 add     ecx, 4
.text:004144D0                 push    ecx             ; void **
.text:004144D1                 call    ?AfxDeleteObject@@YGXPAPAX@Z ; AfxDeleteObject(void * *)
.text:004144D6                 mov     ecx, [ebp+var_C]
.text:004144D9                 mov     large fs:0, ecx
.text:004144E0                 leave
.text:004144E1                 retn
.text:004144E1 ;   } // starts at 4144C9
.text:004144E1 ; } // starts at 4144B5
.text:004144E1 sub_4144B5      endp
.text:004144E1
```

- x86 架构程序保护后，如图所示：

```
.text:004144B5
.text:004144B5 loc_4144B5:                             ; CODE XREF: sub_412E0B+3↑p
.text:004144B5                 call    sub_466CF0
.text:004144BA                 out     6Ch, eax
.text:004144BC                 sub     [esi+50h], ebp
.text:004144BF                 call    sub_466D94
.text:004144C4                 push    ss
.text:004144C5                 adc     dword ptr [edi-37245E23h], 64h
.text:004144C5 ; ---------------------------------------------------------------------------
.text:004144CC                 db 0F7h
.text:004144CD ; ---------------------------------------------------------------------------
.text:004144CD
.text:004144CD loc_4144CD:                             ; CODE XREF: .text:0041453D↓j
.text:004144CD                 dec     ebx
.text:004144CE                 or      ebp, edi
.text:004144D0                 adc     bl, bh
.text:004144D2                 movsb
.text:004144D3                 cmc
.text:004144D4                 mov     [ebx-18h], edx
.text:004144D7                 test    [ecx], ebp
.text:004144D9                 add     eax, 1102B200h
.text:004144DE                 jle     short loc_41447C
.text:004144E0                 mov     cl, 10h
.text:004144E2
```

- ARM 架构程序保护前，如图所示：

```
.text:00076E9C
.text:00076E9C ; android::register_android_database_CursorWindow(_JNIEnv *)
.text:00076E9C                 EXPORT _ZN7android38register_android_database_CursorWindowEP7_JNIEnv
.text:00076E9C _ZN7android38register_android_database_CursorWindowEP7_JNIEnv
.text:00076E9C                                         ; CODE XREF: android::register_android_database
.text:00076E9C                                         ; DATA XREF: LOAD:0001B730↑o ...
.text:00076E9C ; __unwind {
.text:00076E9C                 PUSH            {R3-R7,LR}
.text:00076E9E                 MOV             R4, R0
.text:00076EA0                 LDR             R6, =(aAndroidDatabas - 0x76EA8)
.text:00076EA2                 LDR             R3, [R0]
.text:00076EA4                 ADD             R6, PC  ; "android/database/CharArrayBuffer"
.text:00076EA6                 LDR             R2, [R3,#0x18]
.text:00076EA8                 MOV             R1, R6
.text:00076EAA                 BLX             R2
.text:00076EAC                 MOV             R5, R0
.text:00076EAE                 CBNZ            R0, loc_76EBE
.text:00076EB0                 LDR             R0, =(aClazzNull - 0x76EBA)
.text:00076EB2                 LDR             R1, =(aCursorwindow - 0x76EBC)
.text:00076EB4                 LDR             R2, =(aUnableToFindCl - 0x76EBE)
.text:00076EB6                 ADD             R0, PC  ; "clazz == NULL"
.text:00076EB8                 ADD             R1, PC  ; "CursorWindow"
.text:00076EBA                 ADD             R2, PC  ; "Unable to find class %s"
.text:00076EBC                 B               loc_76EE2
.text:00076EBE ; ---------------------------------------------------------------------------
.text:00076EBE
.text:00076EBE loc_76EBE                               ; CODE XREF: android::register_android_database
.text:00076EBE                 LDR             R6, =(aData - 0x76EC8)
.text:00076EC0                 MOV             R1, R5
.text:00076EC2                 LDR             R0, [R4]
.text:00076EC4                 ADD             R6, PC  ; "data"
.text:00076EC6                 LDR             R3, =(aC - 0x76ED4)
.text:00076EC8                 LDR.W           R7, [R0,#0x178]
.text:00076ECC                 MOV             R2, R6
.text:00076ECE                 MOV             R0, R4
.text:00076ED0                 ADD             R3, PC  ; "[C"
.text:00076ED2                 BLX             R7
.text:00076ED4                 CBNZ            R0, loc_76EE8
.text:00076ED6                 LDR             R0, =(aResNull - 0x76EE0)
.text:00076ED8                 LDR             R1, =(aCursorwindow - 0x76EE2)
.text:00076EDA                 LDR             R2, =(aUnableToFindSt - 0x76EE4)
.text:00076EDC                 ADD             R0, PC  ; "res == NULL"
.text:00076EDE                 ADD             R1, PC  ; "CursorWindow"
.text:00076EE0                 ADD             R2, PC  ; "Unable to find static field %s"
.text:00076EE2
.text:00076EE2 loc_76EE2                               ; CODE XREF: android::register_android_database
.text:00076EE2                                         ; android::register_android_database_CursorWind
.text:00076EE2                 MOV             R3, R6
.text:00076EE4                 BLX             __android_log_assert
```

00067EBE 00076EBE: android::register_android_database_CursorWindow(_JNIEnv *):loc_76EBE (Synchronized with Hex View-1)

- ARM 架构程序保护后，如图所示：

```
.text:00098E9C
.text:00098E9C ; android::register_android_database_CursorWindow(_JNIEnv *)
.text:00098E9C                 EXPORT _ZN7android38register_android_database_CursorWindowEP7_JNIEnv
.text:00098E9C _ZN7android38register_android_database_CursorWindowEP7_JNIEnv
.text:00098E9C                                         ; DATA XREF: LOAD:0000C8E0↑o
.text:00098E9C ; __unwind {
.text:00098E9C                 B.W             _ZN7android38register_android_database_CursorWindowEP7
.text:00098E9C ; End of function android::register_android_database_CursorWindow(_JNIEnv *)
.text:00098E9C
.text:00098EA0 ; ---------------------------------------------------------------------------
.text:00098EA0                 SUBS            R6, #0xF6
.text:00098EA2                 STRB            R4, [R4,#0x1B]
.text:00098EA4
.text:00098EA4 loc_98EA4                               ; CODE XREF: .text:0003ABE8↑j
.text:00098EA4                 ADD             R6, PC
.text:00098EA6                 B.W             loc_3A0CC
.text:00098EAA ; ---------------------------------------------------------------------------
.text:00098EAA                 BLX             R2
.text:00098EAC                 MOV             R5, R0
.text:00098EAE                 CBNZ            R0, loc_98EBE
.text:00098EB0                 B.W             loc_3A148
.text:00098EB4 ; ---------------------------------------------------------------------------
.text:00098EB4                 B               loc_98BE6
.text:00098EB6 ; ---------------------------------------------------------------------------
.text:00098EB6                 ADD             R0, PC
.text:00098EB8                 ADD             R1, PC
.text:00098EBA                 ADD             R2, PC
.text:00098EBC                 B               loc_98EE2
.text:00098EBE ; ---------------------------------------------------------------------------
.text:00098EBE
.text:00098EBE loc_98EBE                               ; CODE XREF: .text:00098EAE↑j
.text:00098EBE                 B.W             loc_3A1C4
.text:00098EBE ; ---------------------------------------------------------------------------
.text:00098EC2                 DCW 0xC7B2
.text:00098EC4                 DCD 0xF7A1447E, 0x5A55B9B9, 0x20EE33F4, 0x47B8447B, 0xF7A1B940
.text:00098EC4                 DCD 0x271DBA01, 0x44794478
.text:00098EE0 ; ---------------------------------------------------------------------------
.text:00098EE0                 ADD             R2, PC
.text:00098EE2
.text:00098EE2 loc_98EE2                               ; CODE XREF: .text:00098EBC↑j
.text:00098EE2                 MOV             R3, R6
.text:00098EE4                 BLX             sub_713B4
.text:00098EE8                 B.W             loc_3A368
.text:00098EE8 ; ---------------------------------------------------------------------------
.text:00098EEC                 DCD 0x447F781B, 0xF7A1447E, 0xD581BA99, 0xA4E47EFD, 0x8E01804B
.text:00098EEC                 DCD 0x47E0447B, 0xF7A1B190, 0x64B1BADB, 0xF7A14479, 0xF7A1BB19
.text:00098EEC                 DCD 0x850EBB63, 0x4790233B, 0xF7A1B968, 0x735CBBA7, 0x44794478
.text:00098EEC                 DCD 0xE01B447A, 0xBBE2F7A1, 0x44785ED5, 0x447A4479, 0xF7A1E7D3
.text:00098EEC                 DCD 0xF891BC11, 0x44FED45B, 0x44794A1E, 0xBC4EF7A1, 0x4620447A
.text:00098EEC                 DCD 0xEC54F7D7, 0xBC8CF7A1, 0xBCDAF7A1, 0x4478ABC7, 0x447A4479
.text:00098EEC                 DCD 0xEA26F7D8, 0xBF00BDF8, 0x52DAC, 0x4E242, 0x52BA3
00098E9C 00098E9C: android::register_android_database_CursorWindow(_JNIEnv *) (Synchronized with Hex View-1)
```

代码虚拟化

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。

保护效果图

- 保护前，如图所示：

```
.text:004056F4 start           proc near
.text:004056F4
.text:004056F4 var_68          = dword ptr -68h
.text:004056F4 lpCmdLine       = dword ptr -64h
.text:004056F4 var_60          = dword ptr -60h
.text:004056F4 StartupInfo     = _STARTUPINFOA ptr -5Ch
.text:004056F4 ms_exc          = CPPEH_RECORD ptr -18h
.text:004056F4
.text:004056F4 ; __unwind { // __except_handler3
.text:004056F4                 push    ebp
.text:004056F5                 mov     ebp, esp
.text:004056F7                 push    0FFFFFFFFh
.text:004056F9                 push    offset stru_416AB0
.text:004056FE                 push    offset __except_handler3
.text:00405703                 mov     eax, large fs:0
.text:00405709                 push    eax
.text:0040570A                 mov     large fs:0, esp
.text:00405711                 sub     esp, 58h
.text:00405714                 push    ebx
.text:00405715                 push    esi
.text:00405716                 push    edi
.text:00405717                 mov     [ebp+ms_exc.old_esp], esp
.text:0040571A                 call    ds:GetVersion
.text:00405720                 xor     edx, edx
.text:00405722                 mov     dl, ah
.text:00405724                 mov     dword_41E524, edx
.text:0040572A                 mov     ecx, eax
.text:0040572C                 and     ecx, 0FFh
.text:00405732                 mov     dword_41E520, ecx
.text:00405738                 shl     ecx, 8
.text:0040573B                 add     ecx, edx
.text:0040573D                 mov     dword_41E51C, ecx
.text:00405743                 shr     eax, 10h
.text:00405746                 mov     dword_41E518, eax
.text:0040574B                 push    1
.text:0040574D                 call    __heap_init
.text:00405752                 pop     ecx
.text:00405753                 test    eax, eax
.text:00405755                 jnz     short loc_40575F
.text:00405757                 push    1Ch             ; NumberOfBytesWritten
.text:00405759                 call    _fast_error_exit
```

- 保护后，如图所示：

```
.text:004056F4 ; ---------------------------------------------------------------------------
.text:004056F4                 push    0
.text:004056F9                 jmp     loc_420000
.text:004056F9 ; ---------------------------------------------------------------------------
.text:004056FE                 db 40h, 42h, 65h, 45h, 39h, 2 dup(0C6h), 14h, 92h, 9, 4Dh
.text:004056FE                 db 0AFh, 0AEh, 1Ch, 0B8h, 56h, 8, 4Ch, 38h, 7Dh, 0Dh, 0EAh
.text:004056FE                 db 0F3h, 8Ch, 19h, 0ACh, 12h, 0C8h, 62h, 4, 36h, 0Ah, 35h
.text:004056FE                 db 7Ah, 7Fh, 0F8h, 79h, 0E7h, 6, 62h, 4Eh, 51h, 0Ah, 0AAh
.text:004056FE                 db 0A4h, 0DAh, 1, 0EEh, 8Ah, 0A3h, 2Eh, 3, 0C1h, 69h, 0D0h
.text:004056FE                 db 0B2h, 61h, 6Dh, 75h, 4Dh, 81h, 92h, 87h, 58h, 0BCh
.text:004056FE                 db 69h, 33h, 0E7h, 8Bh, 0C9h, 69h, 0C5h, 0BAh, 0CDh, 0C1h
.text:004056FE                 db 0A7h, 45h, 66h, 8Fh, 0FFh, 71h, 0CFh, 9Dh, 40h, 0B0h
.text:004056FE                 db 90h, 37h, 16h, 0EEh, 7Dh, 42h, 70h, 2 dup(35h), 1Eh
.text:004056FE                 db 64h, 2Eh, 83h, 26h, 0D8h, 75h, 0DDh, 3Fh, 0A1h, 0E2h
.text:004056FE                 db 8Dh, 0A4h, 0F2h, 3, 0A4h, 17h, 5Ch, 49h, 0F8h, 27h
.text:004056FE                 db 76h, 90h, 0CBh, 7, 0BBh, 7, 0AEh, 89h, 61h, 98h, 9Bh
.text:004056FE                 db 86h, 7Fh, 70h, 0F8h, 9Eh, 5Eh, 0FAh, 0D0h, 57h, 0C7h
.text:004056FE                 db 0FCh, 38h, 0AEh, 64h, 0Ch, 85h, 67h, 3Ah, 0B2h, 9Bh
.text:004056FE                 db 7, 0F9h, 0EAh, 0ACh, 0C9h, 0BAh, 8Bh, 67h, 65h, 0D9h
.text:004056FE                 db 0D2h, 96h, 0CDh, 3Dh, 0D9h, 0B7h, 0FEh, 83h, 0F9h, 3Eh
.text:004056FE                 db 0B4h, 0E9h, 82h, 9Bh, 82h, 66h, 0CEh, 49h, 2Dh, 7Fh
.text:004056FE                 db 11h, 8Eh, 0F3h, 0DEh, 0FEh, 72h, 4Bh, 37h, 32h, 38h
.text:004056FE                 db 0ADh, 0B2h, 0EEh, 3Dh, 17h, 62h, 53h, 63h, 3Dh, 28h
.text:004056FE                 db 89h, 0FCh, 0Dh, 34h, 60h, 0ACh, 45h, 68h, 28h, 92h
.text:004056FE                 db 51h, 36h, 0D6h, 86h, 5Dh, 7Ch, 6Ch, 46h, 31h, 43h, 35h
.text:004056FE                 db 5Ch, 0EFh, 0F6h, 0C3h, 76h, 2Eh, 3Ch, 75h, 32h, 63h
.text:004056FE                 db 5Fh, 0A6h, 7, 99h, 57h, 15h, 0E0h, 0FDh, 0F5h, 4Dh
.text:004056FE                 db 7Bh, 0E3h, 65h, 7Fh, 1Dh, 0B7h, 8Bh, 65h, 0E8h, 0FFh
.text:004056FE                 db 75h, 98h, 0E8h, 3Fh, 10h, 2 dup(0)
.text:004057FC
```

自动化保护

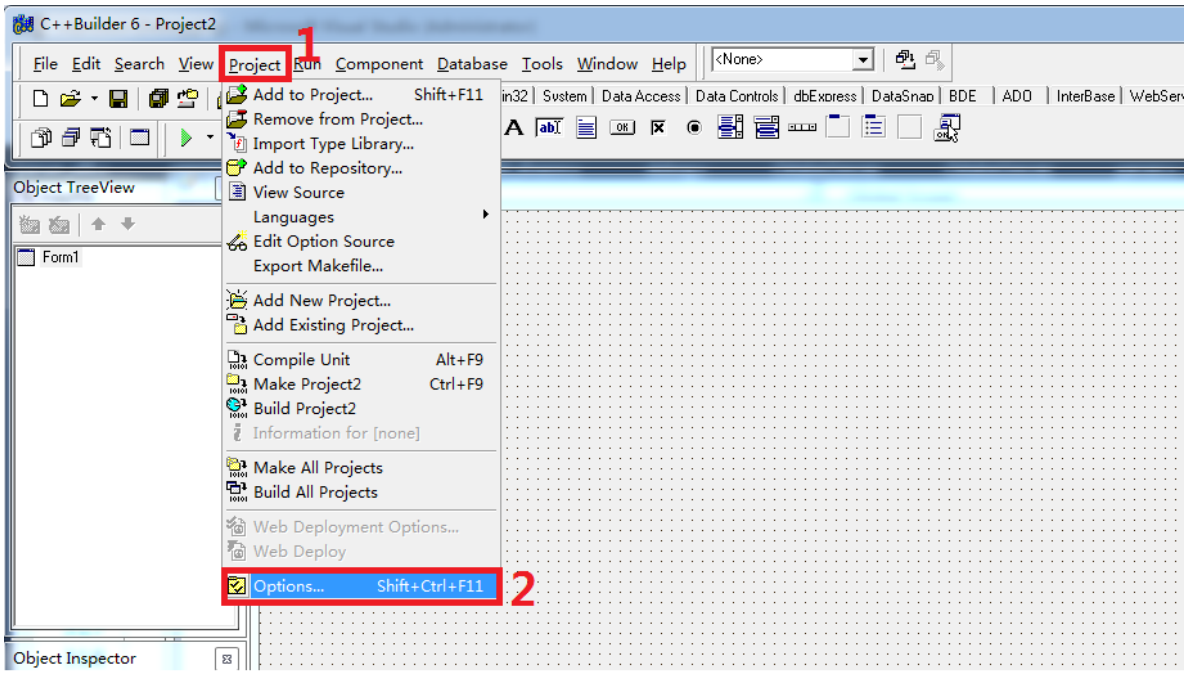
使用 map 文件

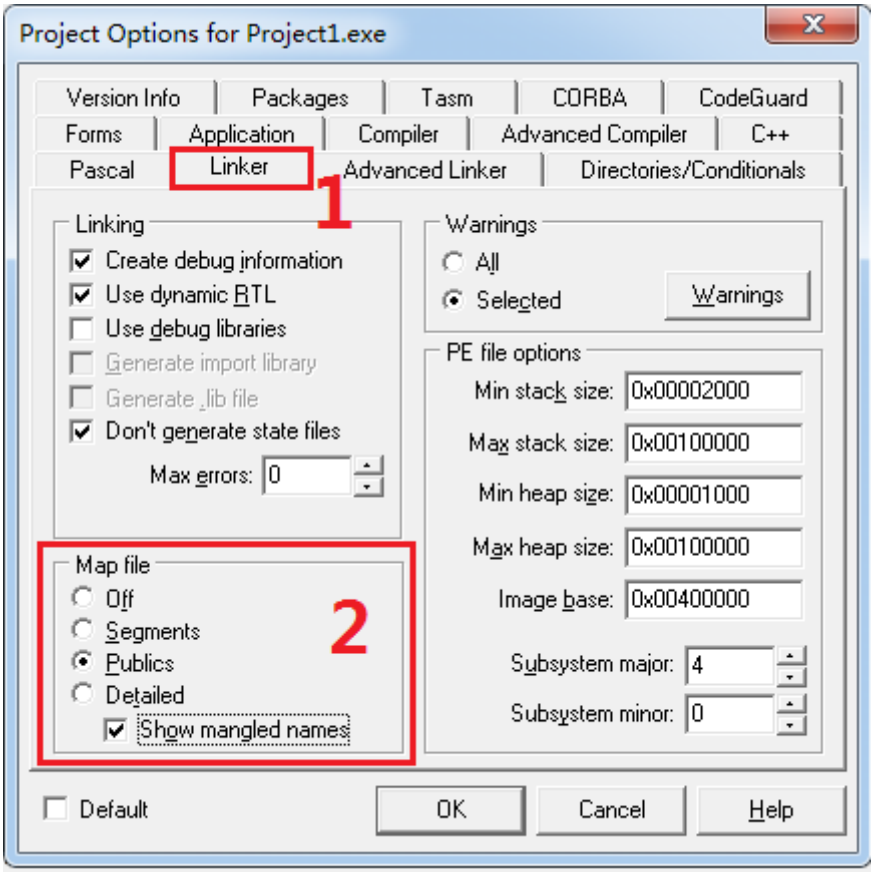
加壳工具解析PE程序时函数是以地址的方式显示，若有map文件则加壳工具解析PE程序时函数是以函数名的方式进行显示。

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。

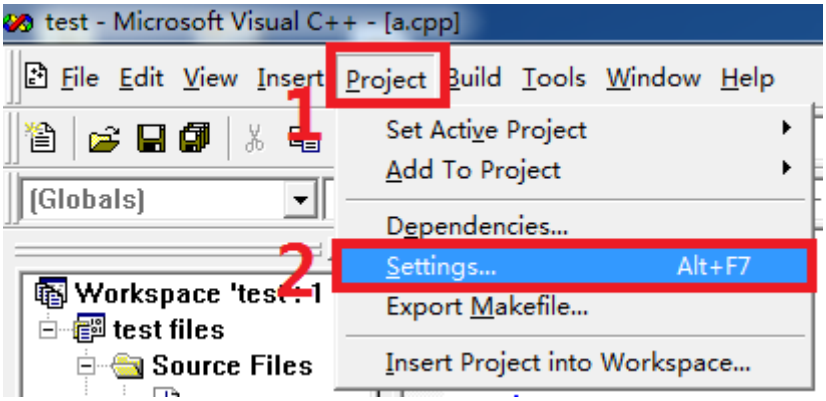
- 工程设置如下图：

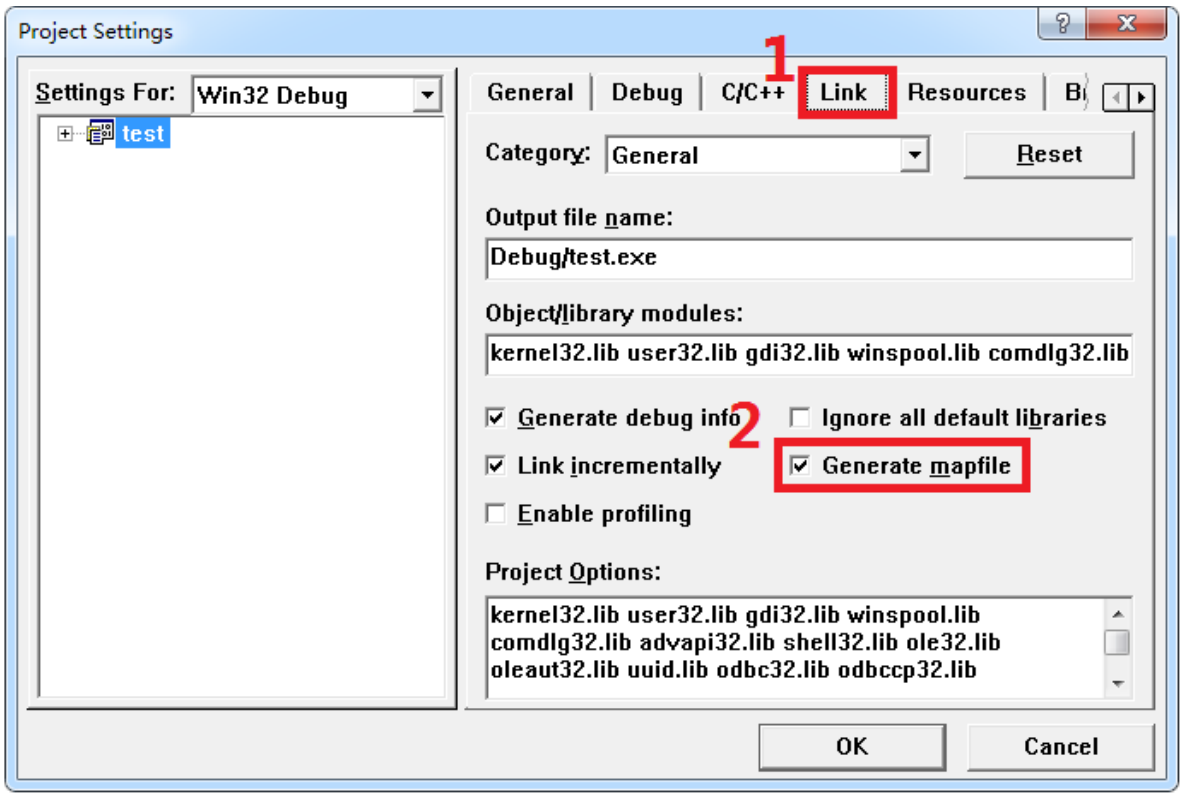




使用VC生成map文件

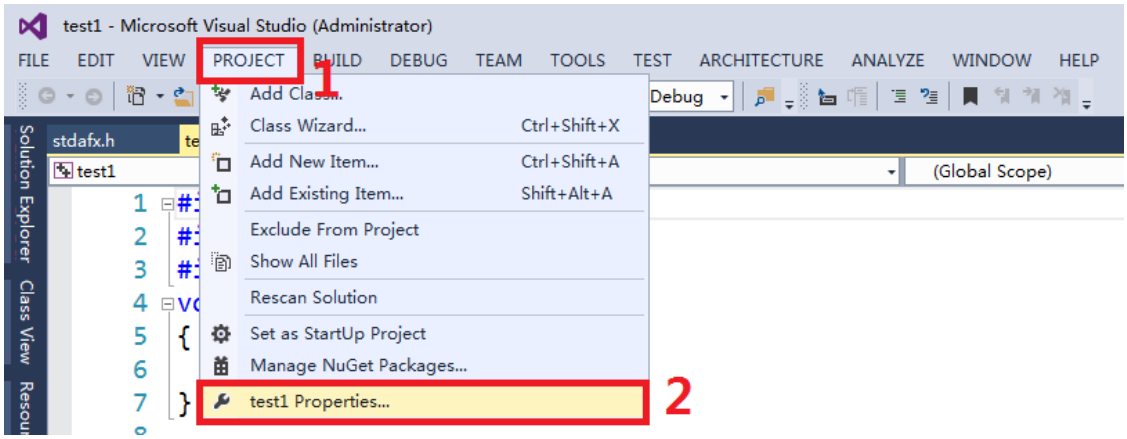
- 工程设置如下图：

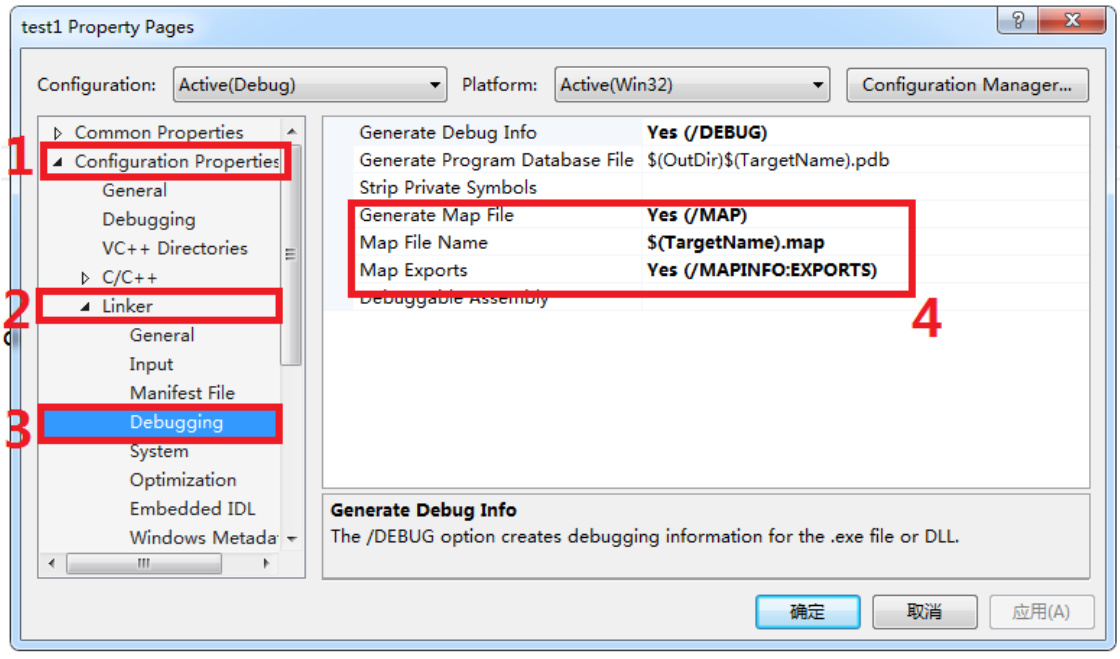




使用 VS 生成 map 文件

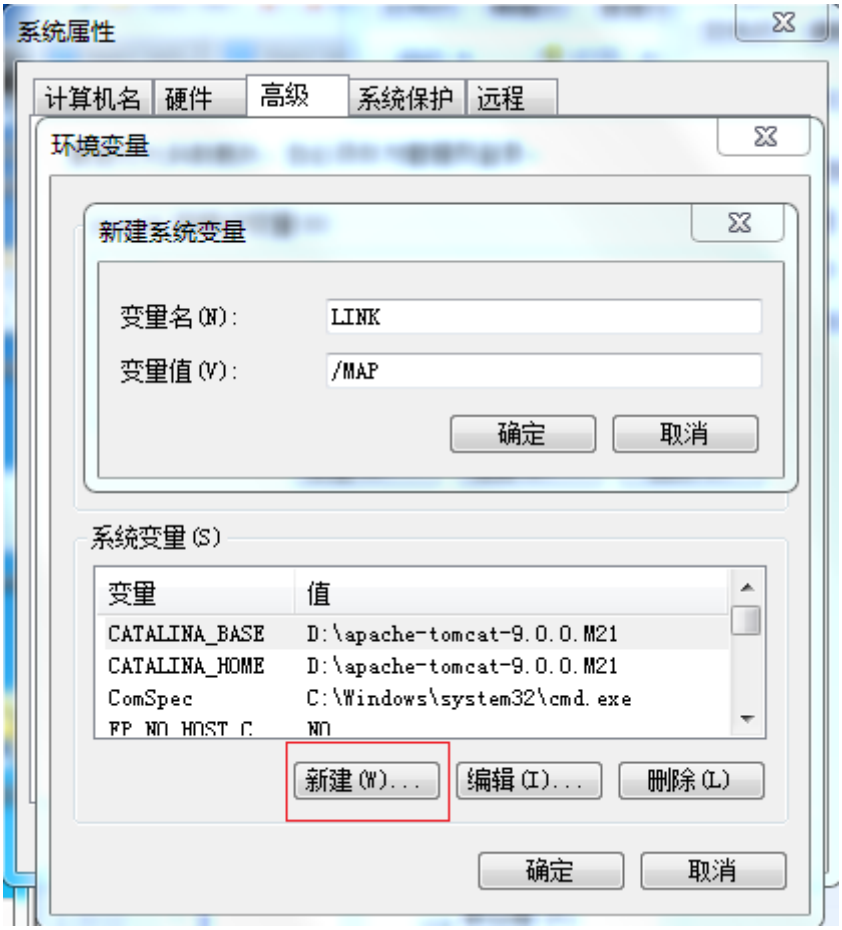
- 工程设置如下图：





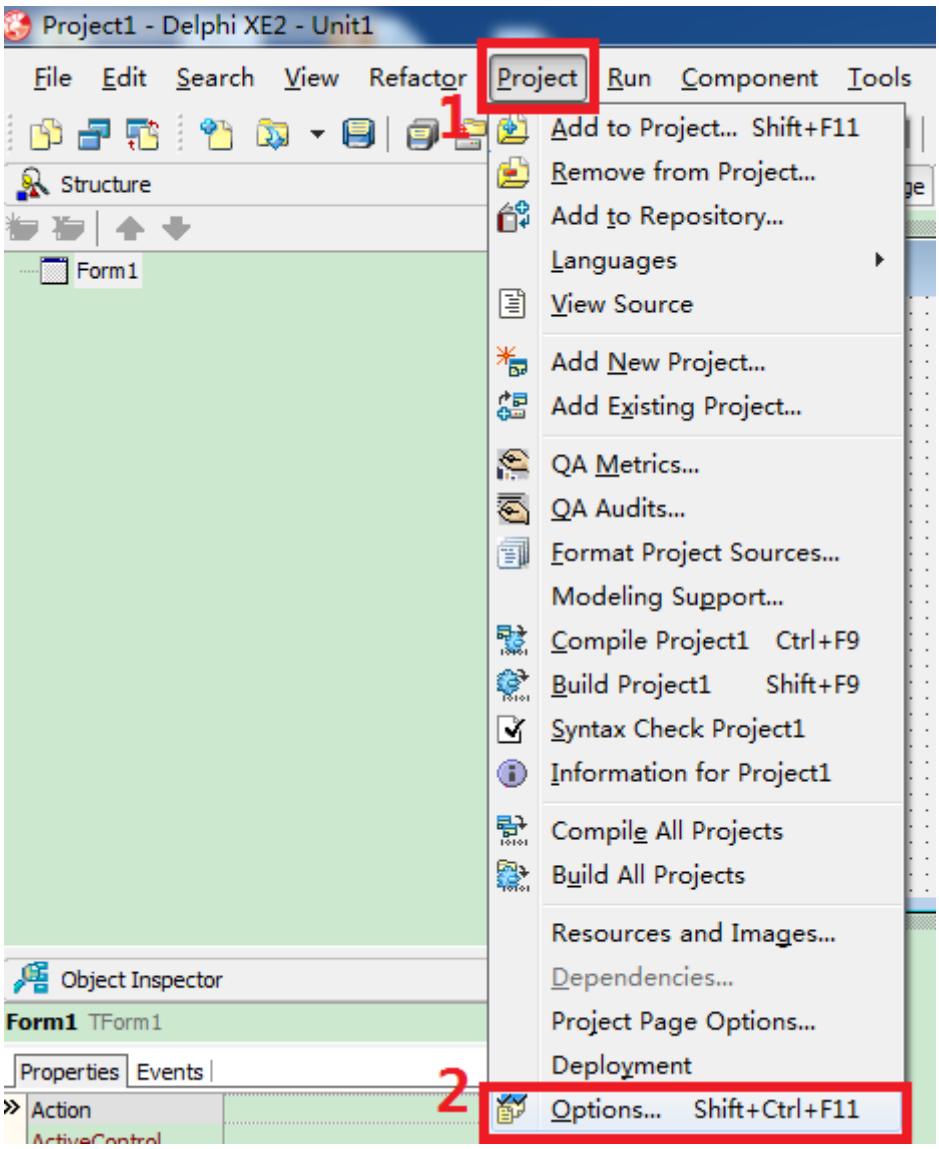
使用 VB6.0 生成 map 文件

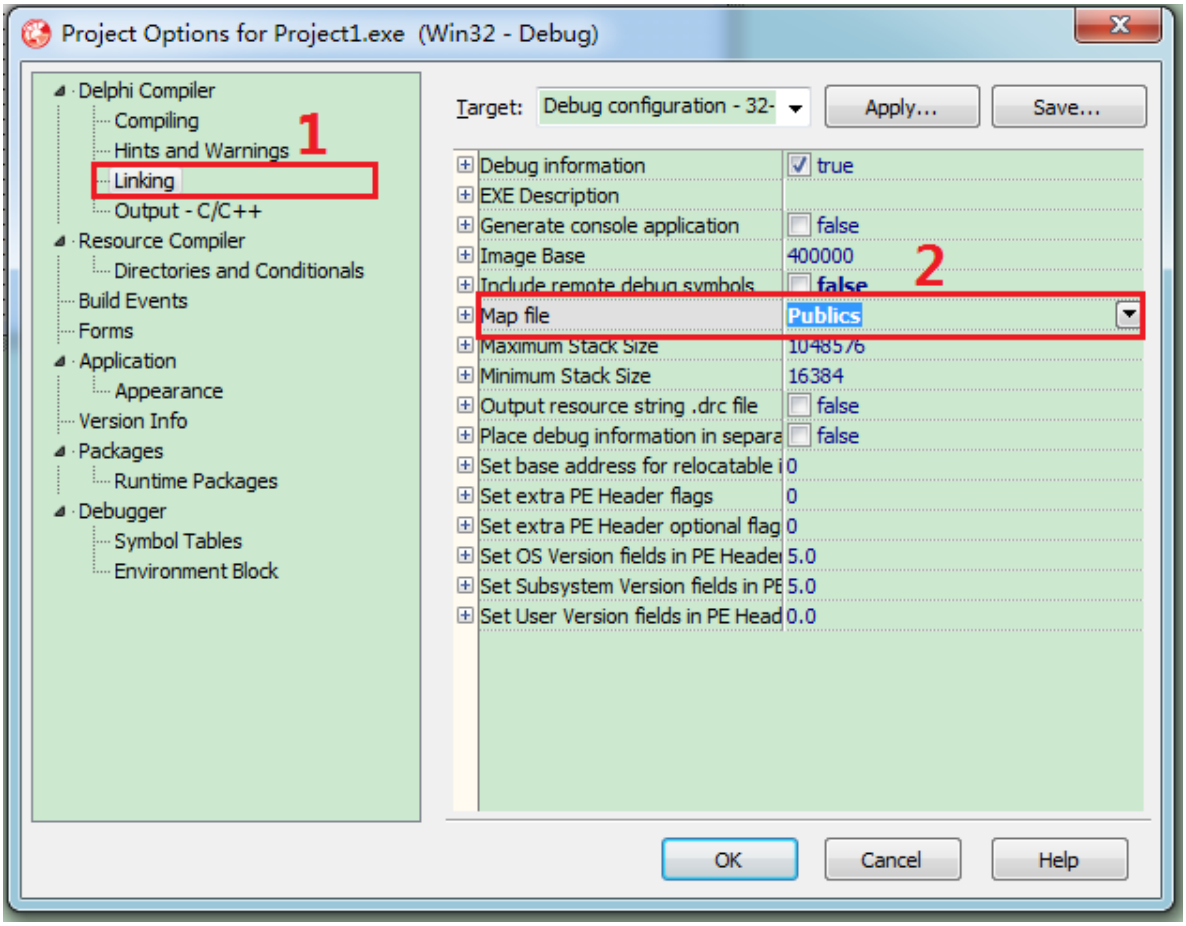
- 工程设置如下图：



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
2
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
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

- 只能静态加载，不支持动态加载dll（即 LoadLibrary 的方式）。
- 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。
- 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。
- 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。
- 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。
- 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）
- 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。
- 目前明确不支持的语言：易语言、Java 程序、Unity3d。
- Begin/End 不支持嵌套使用。
- 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"};

- VBProtectDecrypt 传入缓冲区和传出缓冲区不能是同一个缓冲区。
- 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。
- .Net 程序暂时不支持。

字符串加解密

- 加密的字符串必须是常量。
- 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。
- 字符串解密支持的写法有以下几种：
  - 直接字符串解密：

    > VBDecryptStringA("test_string");

  - 局部静态变量：

    > static const char g_string[] = "test_string";

  - 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";

- 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。
- 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：

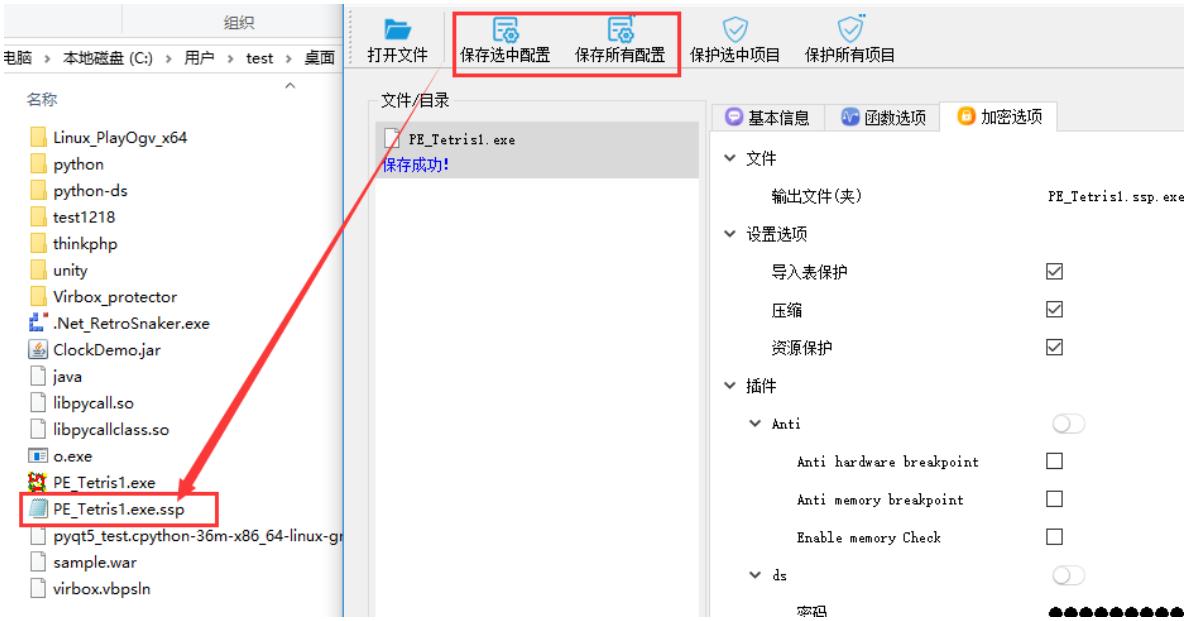
  ```
  const char* a = "test_string";
  const char* b = VBDecryptStringA("test_string");
  printf("a = %s, b = %s\n", a, b);
  ```

  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。



自动生成 ssp 文件

需要联系**深思数盾客服**，我们将单独提供生成工具。

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		

Virbox Protector

Linux命令行

1、普通程序使用方法，以Linux平台程序为例：

- 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.
```

- 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.
```

2、Unity3D程序使用方法

Unity3D作为一个特殊的文件类型，和普通程序的保护方式不同，针对Windows、Linux和macos平台的Unity3D，需要对Unity3D整个目录进行保护；针对Android平台的Unity3D，需要对Unity3D的apk进行保护。**以Linux Unity3D为例：**

- 使用Virbox Protector界面工具生成配置文件（可选）
- 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
- 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 -u3d -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/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.
```

- 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.
```

Windows命令行

1、普通程序使用方法，以 Linux 平台程序为例：

- 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.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.
```

- 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.
```

2、Unity3D 程序使用方法

Unity3D 作为一个特殊的文件类型，和普通程序的保护方式不同，针对 Windows、Linux 和 macOS 平台的Unity3D，需要对 Unity3D 整个目录进行保护；针对 Android 平台的 Unity3D，需要对 Unity3D 的 apk 进行保护。**以 Android Unity3D 为例：**

- 使用 Virbox Protector 界面工具生成配置文件（可选）
- 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
- 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.exe的路径 需要被保护的程序路径 -u3d -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\angrybots5.5.3.apk -u3d -o C:\Users\test\Desktop\sample\ssp.angrybots5.5.3.apk
SenseShield Virbox [version: 1.4.2.10236]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


protect unity3d angrybots5.5.3.apk ...
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\angrybots5.5.3.apk -u3d -o C:\Users\test\Desktop\sample\ssp.angrybots5.5.3.apk
SenseShield Virbox [version: 1.4.2.10236]
Copyright(c) SenseShield Technology Co., Ltd.  All rights reserved.


protect unity3d angrybots5.5.3.apk ...
Succeed.
```

.NET 程序选项

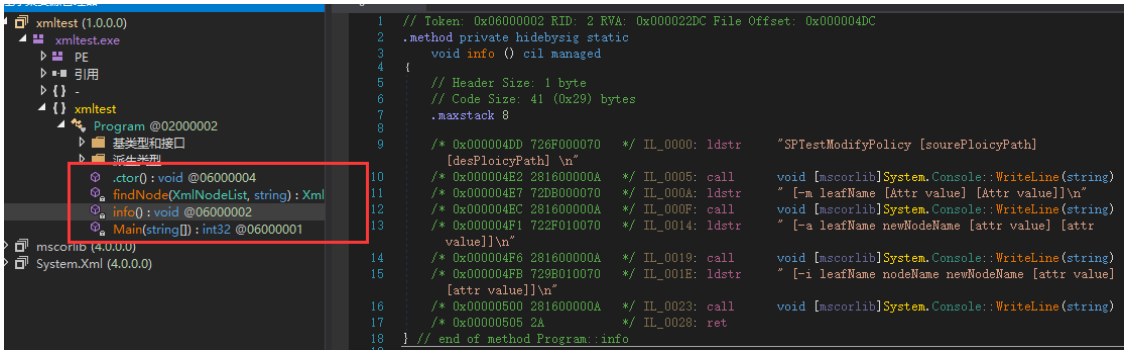
基础保护

名称混淆（.NET）

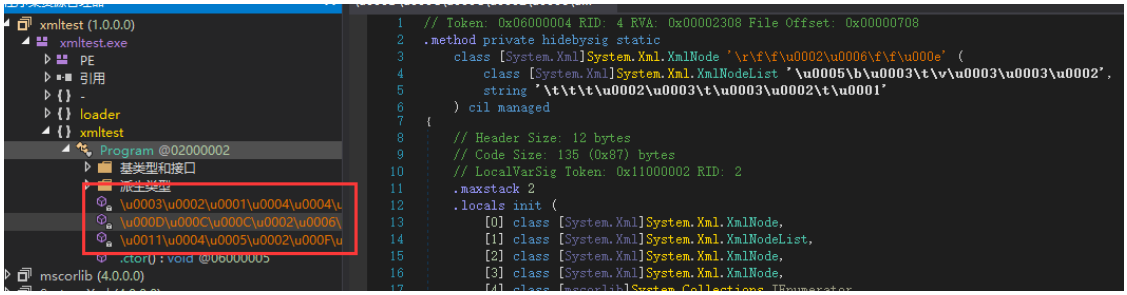
将 .net 的方法名类名使用随机字符串重新命名，导出和外部的名称不会改变。

保护效果图

- 保护前，如图所示：



- 保护后，如图所示：



压缩

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。

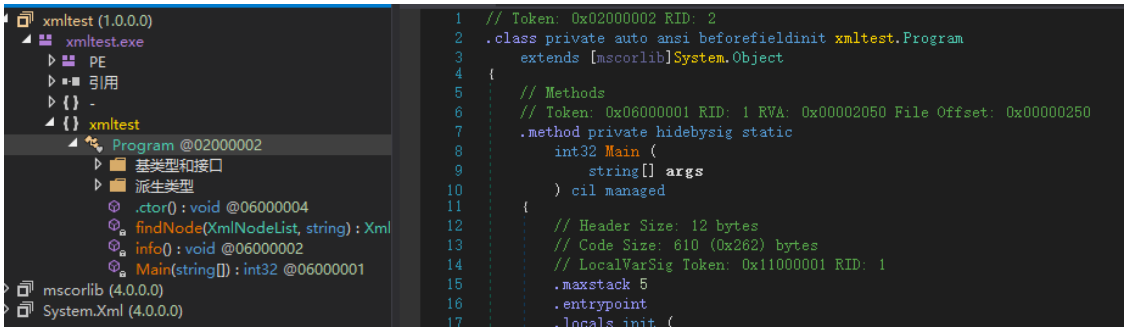
  2、运行效率高，仅在程序被加载时轻微的性能损失。

- **缺点**

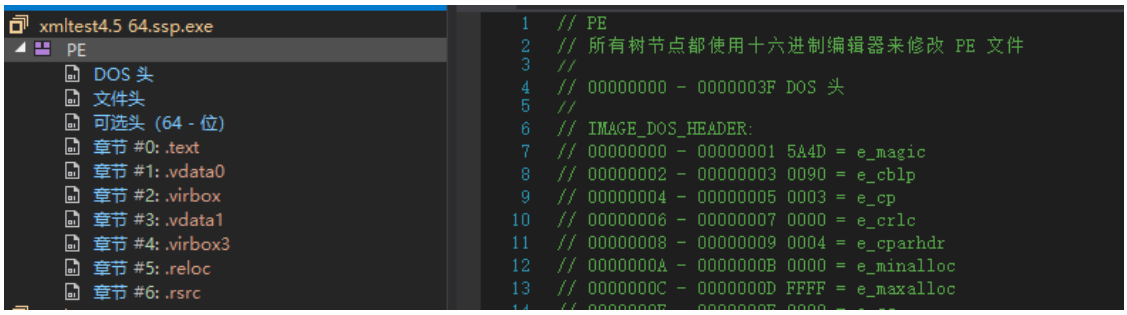
  1、壳代码执行完毕后，代码段与数据段会还原，可以被 Dump。

保护效果图

- 保护前，如图所示：



- 保护后，如图所示：



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

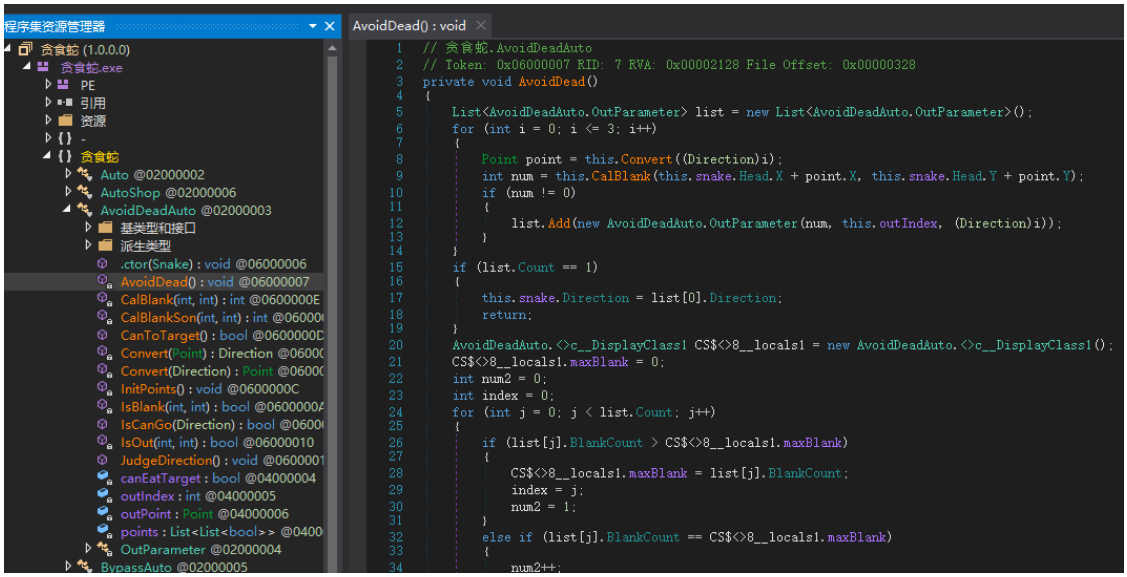
  1、运行效率高，几乎没有性能损失。

- 缺点

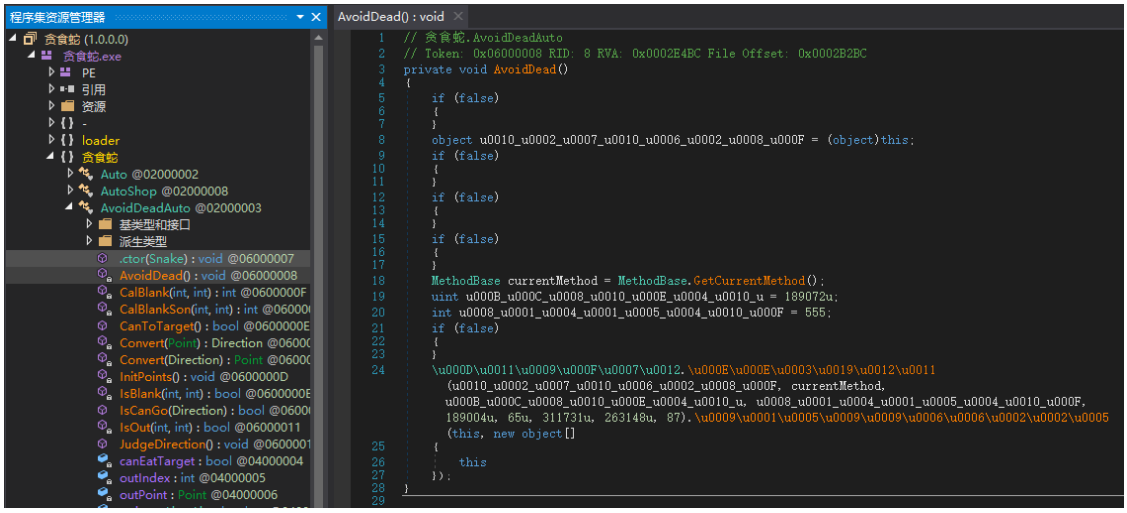
  1、方法执行后会解密，解密之后容易被分析。

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：

- 值类型（及其继承类）的非静态方法 System.ValueType
- 泛型方法暂不支持
- C++ .net不支持
- 递归调用不支持
- 可变参数不支持
- 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**

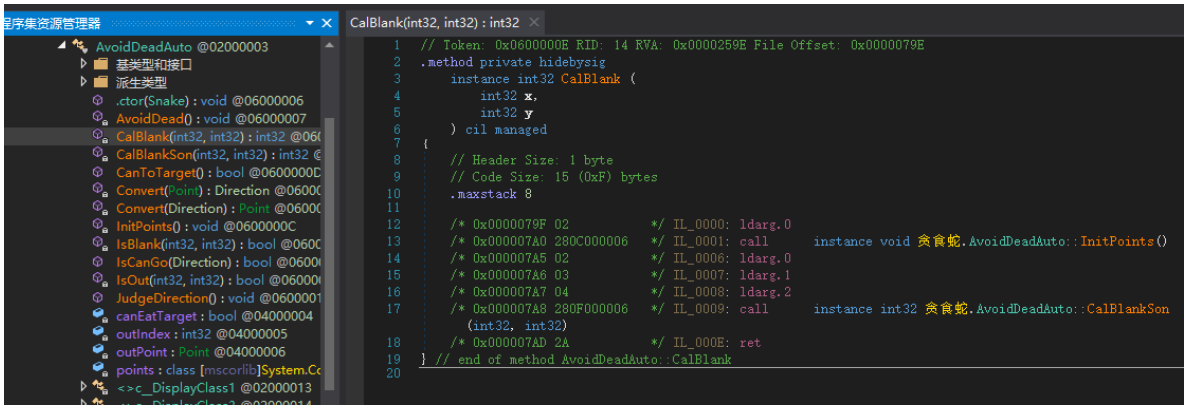
  1、防反编译。

- **缺点**

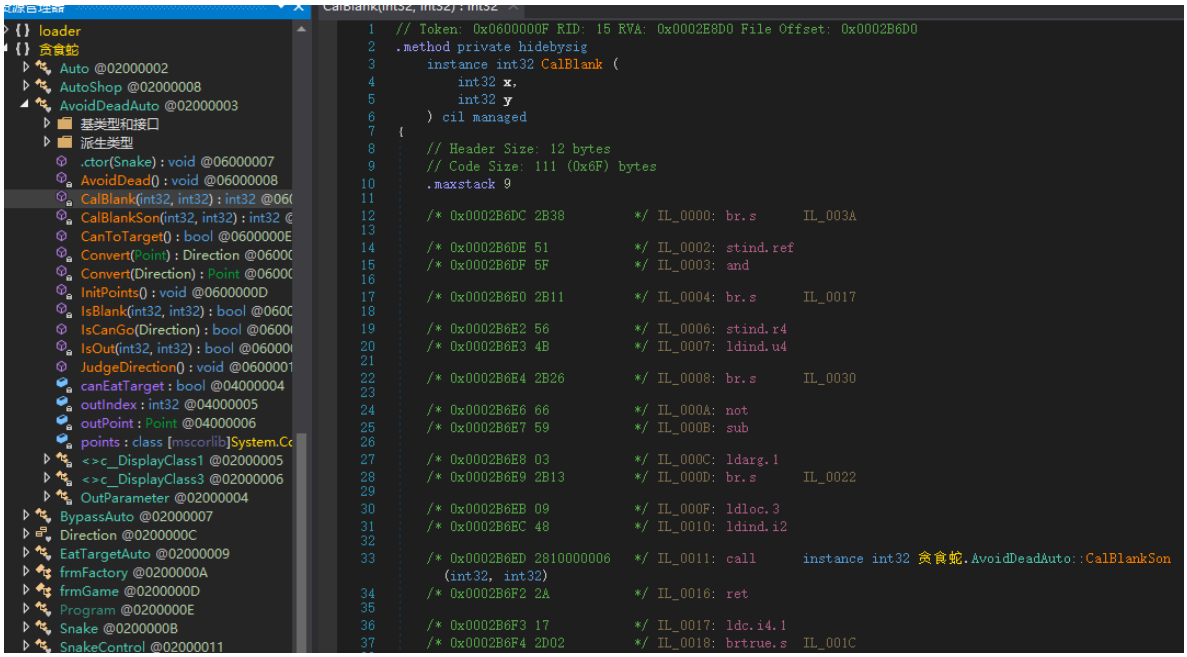
  1、运行效率略有损失，保护强度不高。

保护效果图

- 保护前，如图所示：



- 保护后，如图所示：



Unity3D 程序保护

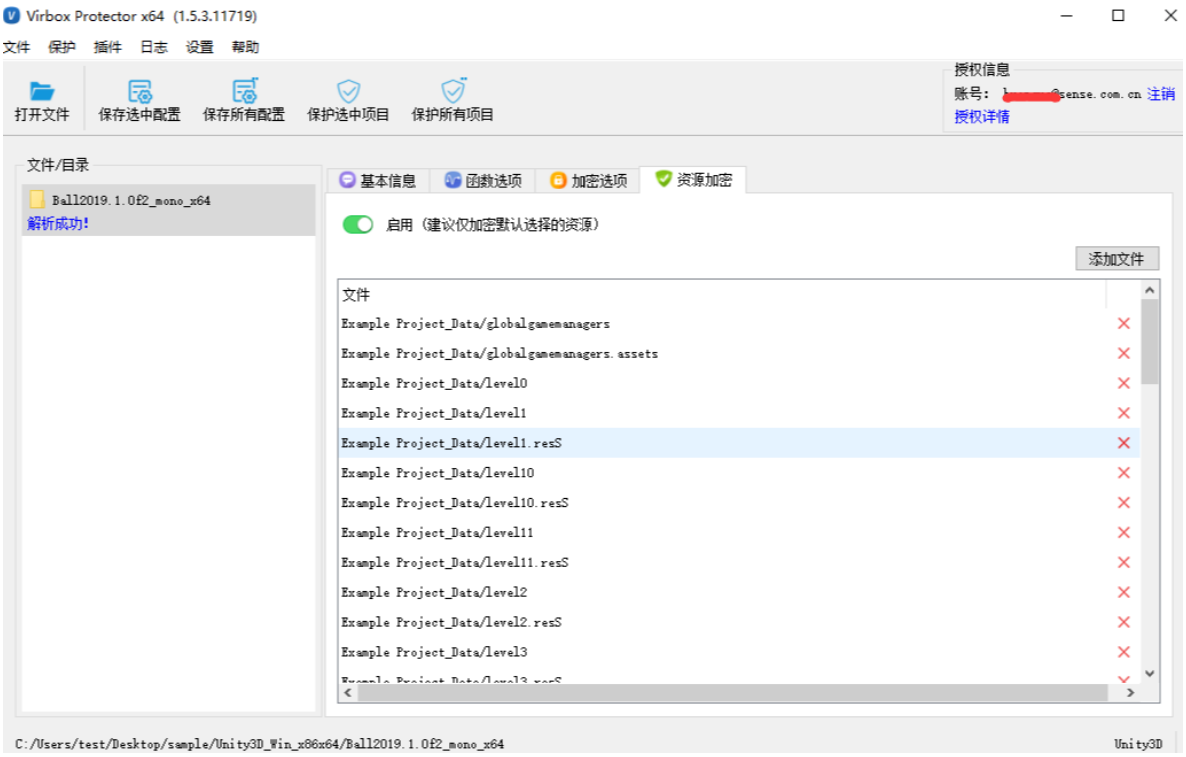
功能

1、对 Unity3D 脚本 C# 代码进行加密，防止逆向和反编译。

2、可以添加程序集文件，即Managed目录下自主开发的C# 程序集。

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。



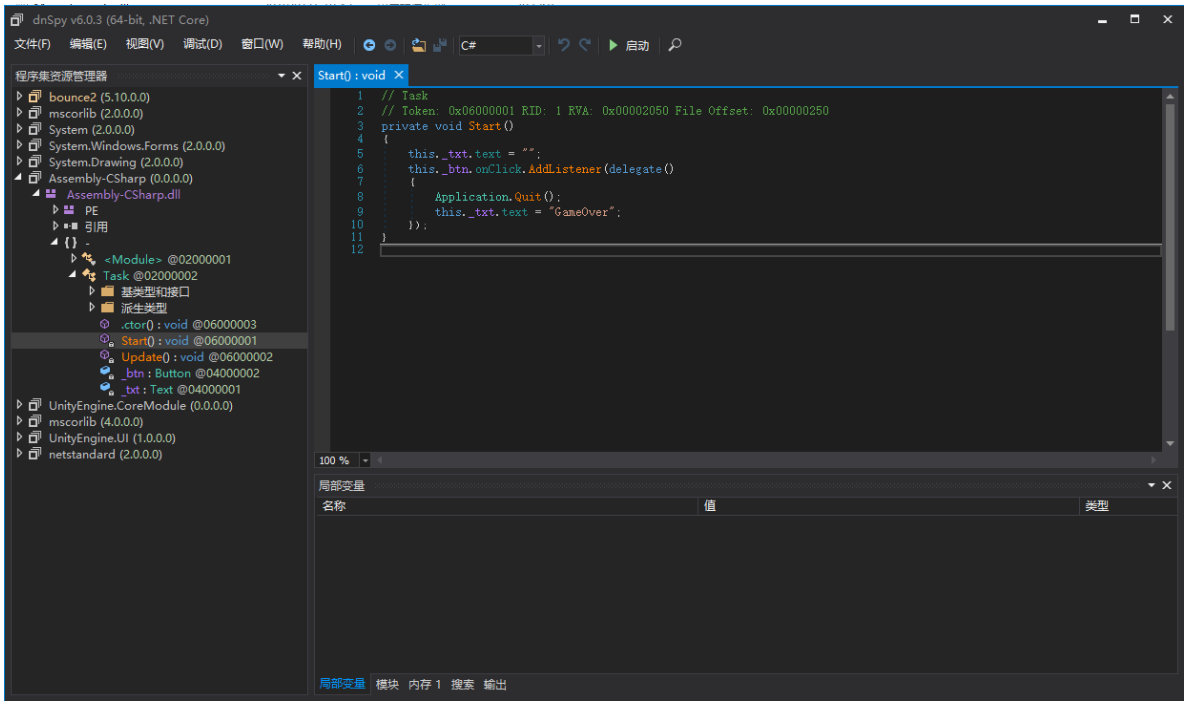
保护原理

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。

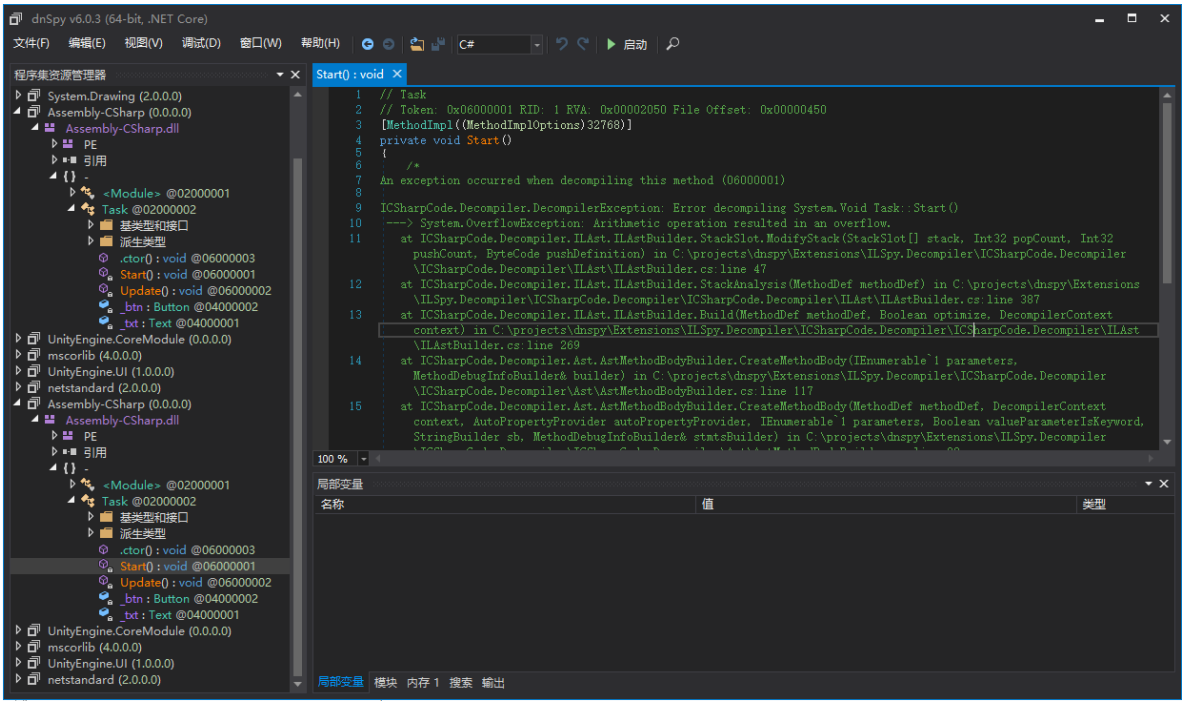
保护效果图

1、下图为 Assembly-CSharp*.dll 脚本文件的保护效果图。

- 保护前，如图所示：

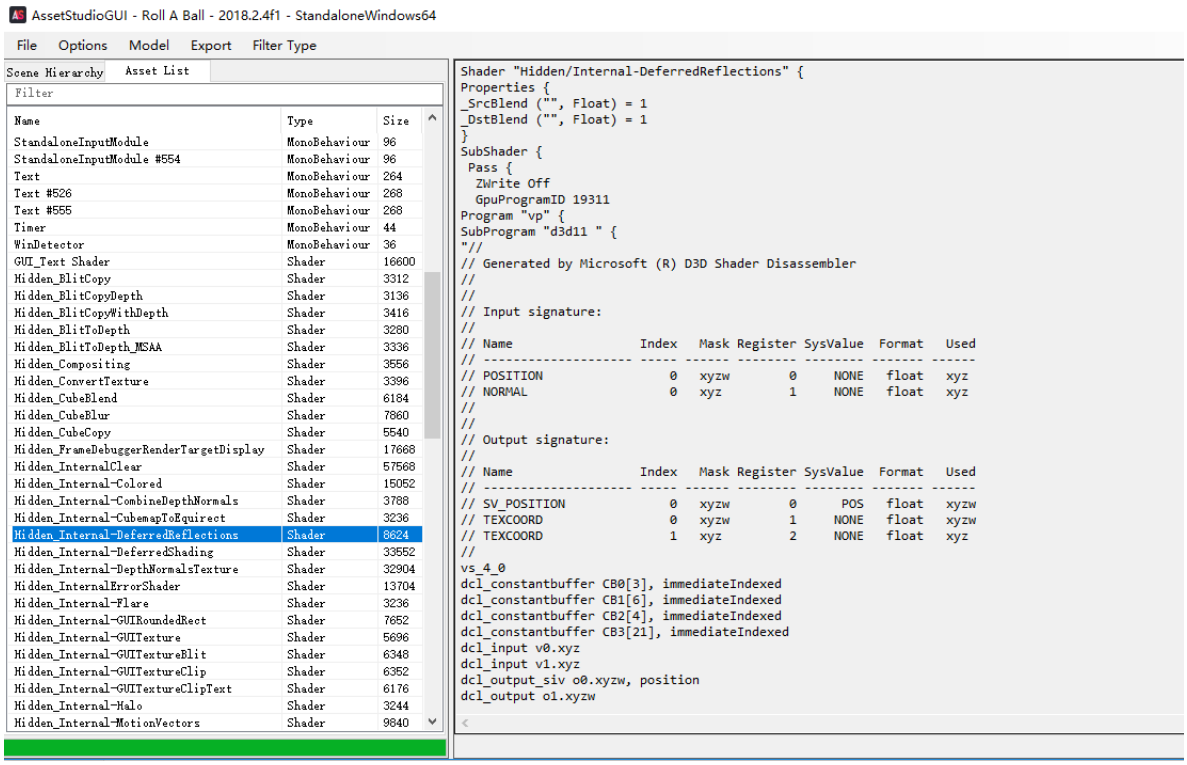


- 保护后，如图所示：

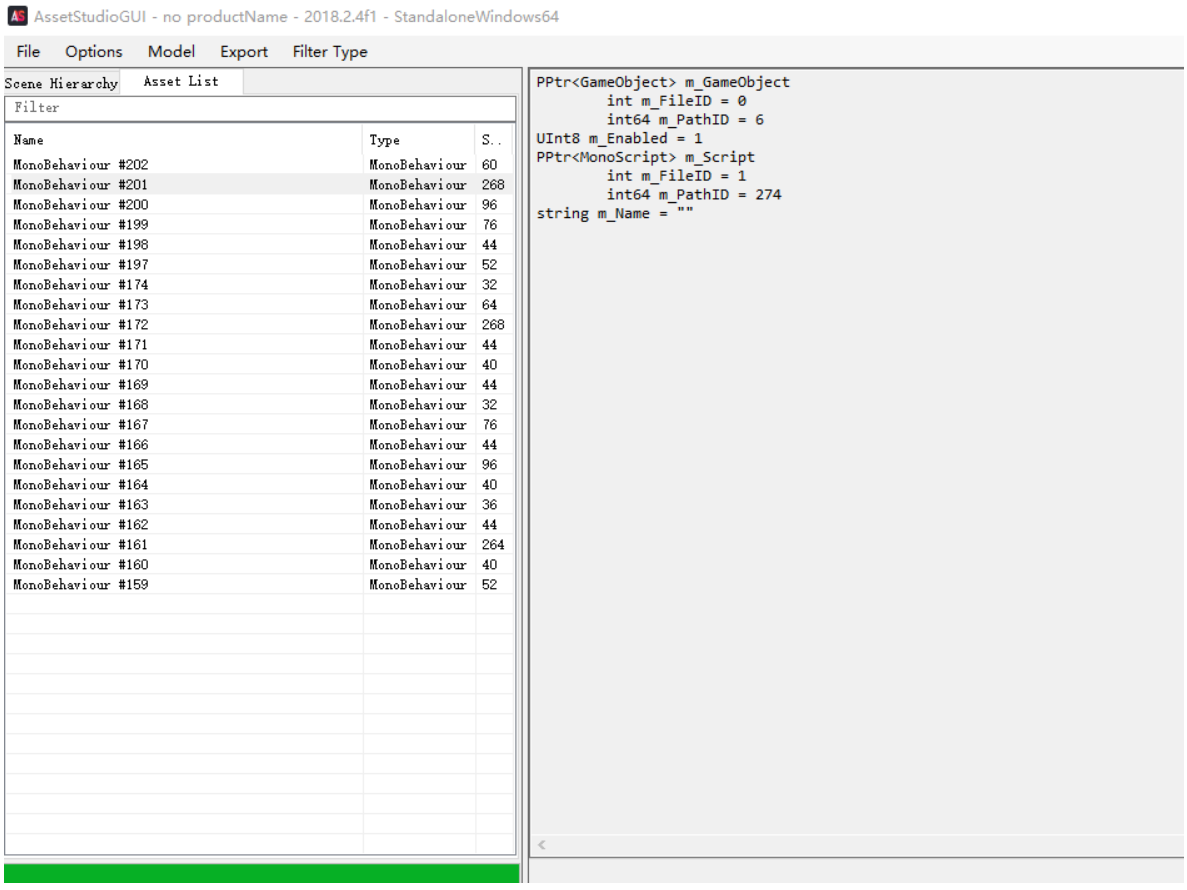


2、Unity3D资源文件的保护效果，可以使用反编译工具(例：AssetStudio)对Unity3D资源进行反编译查看一下效果。

- 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程序保护

普通apk程序

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。

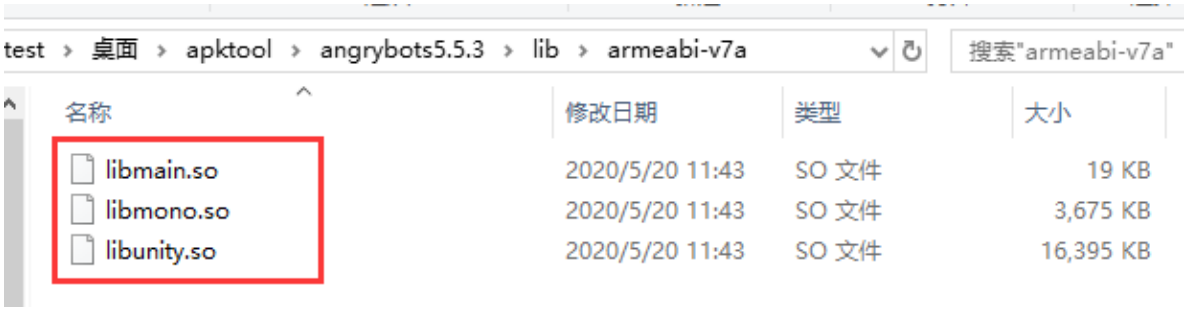
2、保护完成后，对文件夹重新打包签名安装即可。

Android Unity3D

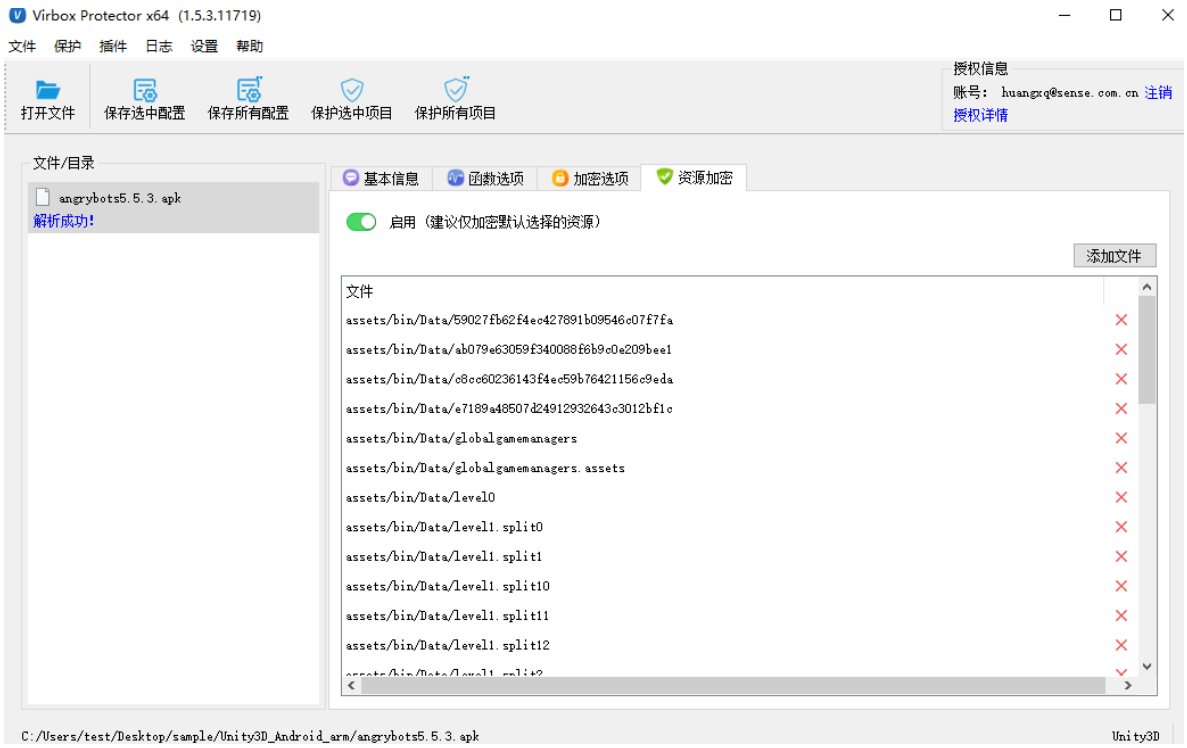
针对Unity3D编译选项时选择的mono和IL2CPP格式，两种保护方式不同。

- mono格式

1、先将Android Unity3D apk解压，查看lib库目录，如图所示：



2、若是lib目录下含有libmono.so库，那么说明Unity3D编译选项时选择的是mono，此时需要对apk整个目录进行加壳。如图所示：

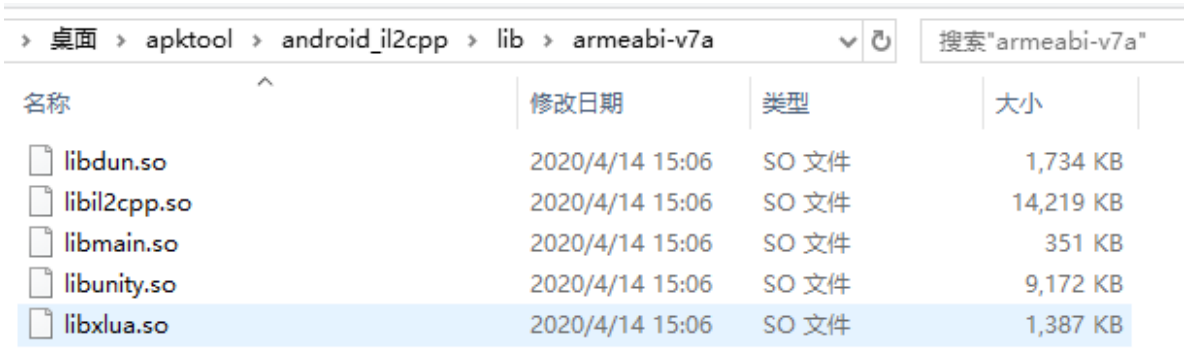


3、可以直接在资源加密选项处点开启用按钮，可以对Unity3D中的资源文件进行加密保护。

4、加壳成功后会生成ssp.apk，然后重新对加壳后的ssp.apk进行签名打包，才能正常安装。

- IL2CPP格式

1、先将Android Unity3D apk解压，查看lib库目录，如图所示：



2、若是lib目录下含有libil2cpp.so库，那么说明Unity3D编译选项时选择的是IL2CPP，此时需要对lib目录下的so库进行保护。

3、保护完成后，对文件夹重新打包签名安装即可。

Java 程序保护

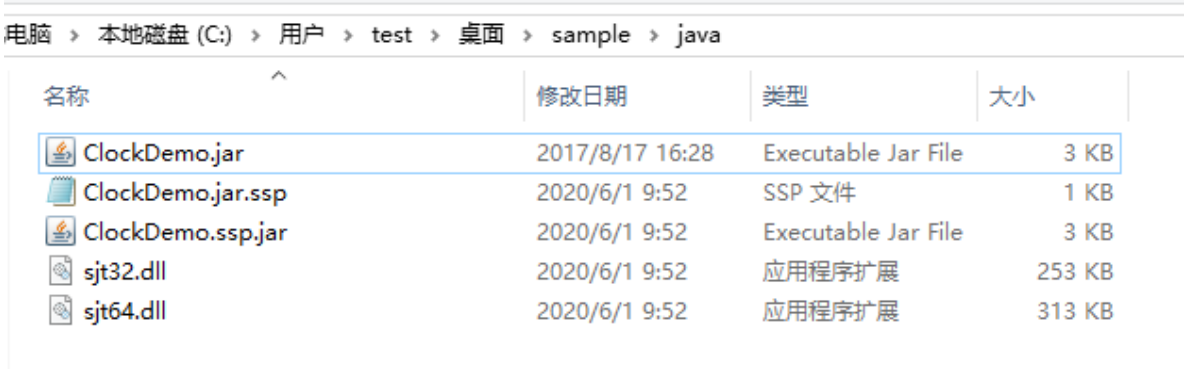
如何对 jar 包保护

使用 Virbox Protector 工具可以直接对未使用框架的 jar 包进行加密保护。

【注意】若使用 Java 框架的 jar包，请使用 DSProtector 进行保护，具体操作方法和使用说明请参考此文档同目录下的《DS.pdf》。

操作流程

- 将 jar包拖入工具中，直接进行保护。
- 保护成功后会生成文件和 sjt 插件。



- 运行加壳后的程序（两种方法，选其一）

  1、-agentpath:sjt64.dll，指定 sjt64.dll 的路径，默认是与 ClockDemo.ssp.jar 同一目录。

  2、Java版本若是 32 位的，使用 sjt32.dll； Java 版本若是 64 位的，使用 sjt64.dll。

  > 命令行：java -agentpath:sjt64.dll -jar ClockDemo.ssp.jar

```
C:\Users\test\Desktop\sample\java>java -agentpath:sjt64.dll -jar ClockDemo.ssp.jar
```

> 命令行：java -agentpath:C:\Users\test\Desktop\sample\java\sjt64.dll -jar ClockDemo.ssp.jar

```
C:\Users\test\Desktop>java -agentpath:C:\Users\test\Desktop\sample\java\sjt64.dll -jar C:\Users\test\Desktop
\sample\java\ClockDemo.ssp.jar
```

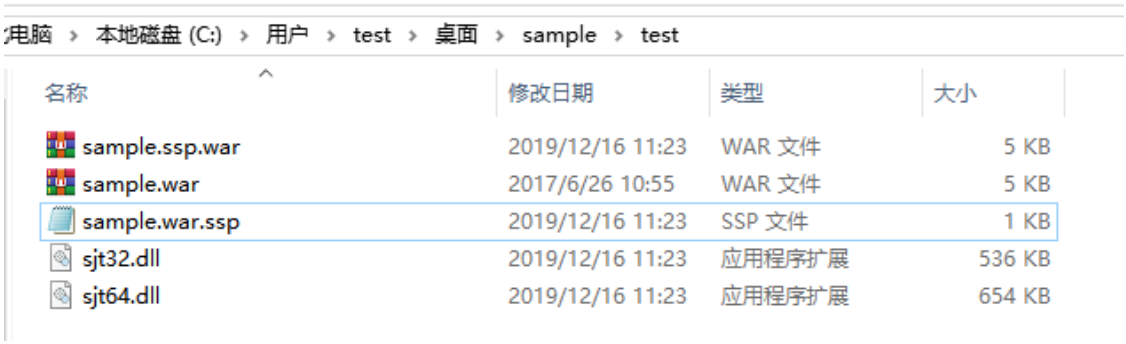
如何对 war 包保护

使用 Virbox Protector 工具可以直接对未使用框架的war包进行加密保护。

【注意】保护后的war包只能在 Windows 平台上运行，若使用框架的 war 包，请使用 DSProtector 进行保护，具体操作方法和使用说明请参考此文档同目录下的《DS.pdf》。

操作流程

- 将war包拖入工具中，直接进行保护。
- 保护成功后会生成文件和sjt插件。



- 若使用Tomcat来运行加密保护后的程序，需要将保护后的 `sample.ssp.war` 放入到 `.\apache-tomcat\webapps` 文件夹中，在 `.\apache-tomcat\bin\catalina.bat` 配置 sjt 库，如图所示：

> 两种配置方法，选其一：
>
> 1、set JAVA_OPTS=-agentpath:sjt64.dll或set JAVA_OPTS=-agentpath:sjt32.dll，将sjt32.dll或sjt64.dll拷贝到jdk\bin或jre\bin目录中。
>
> 2、set JAVA_OPTS=-agentpath:C:\Users\test\Desktop\sample\test\sjt64.dll或set JAVA_OPTS=-agentpath:C:\Users\test\Desktop\sample\test\sjt32.dll，指定sjt库的路径，不需要拷贝。

- 运行 `.\apache-tomcat\bin\startup.bat`，网页启动 `http://localhost:8080/sample.ssp` 正常运行即可。

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

- 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。
- 解决方法：提交 360 认证
  - 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）。
  - 注册并登录 360 开放平台：http://open.soft.360.cn/
  - 在“我的软件”页面中选择提交我的软件，如下图：
  - 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。

  - 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
		加壳工具不支持对spring框架的jar包直接加壳，如果是spring框架的Java程序，请使用资源加密的方式保护
		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
		SDK标签不支持.NET程序
		C#开发的.NET程序或DLL库中含有外部引用或公开的方法，此类程序不能加名称混淆，若选择名称混淆会改变方法名，会使程序某个函数无法正常使用
		.NET 的AnyCPU加压缩后不支持被其他的.NET 模块引用，原因是加压缩后会将.NET程序类型变为PE32
		.NET的DLL没有压缩功能
	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
		导入表：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
		不支持-static编译的ELF格式的程序
		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
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架构程序不支持虚拟化和碎片化

已知问题

- .NET 加壳不支持第三方运行时库，只支持微软标准运行时库。
- 使用命令行加壳时，其目标程序的配置文件必须存在。

- 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。
- 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。
- AutoCAD 的 ARX 插件程序只能选择“远程桌面服务会话消息框”，并且目前只支持 win7 和 server2008 以上的版本。
- 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。
- 暂不支持带有程序集签名(强签名)的程序进行加壳。