《PB混淆加密大师》2012最新版使用引导

http://www.mis2erp.com
http://blog.csdn.net/chengg0769
http://www.mis2erp.com/soft/PBobfuscator.rar

保护PowerBuilder/PocketBuilder编译后的PBD,DLL,PKB文件不被反编译和破解，支持PB5至PB12.5，PKB2.5等众多版本
本文档对该软件做简单的介绍和操作引导

MAIL: chengang0769@gmail.com
chengang0769@21cn.com
MSN: chengang0769@live.cn
QQ: 27-3939-617

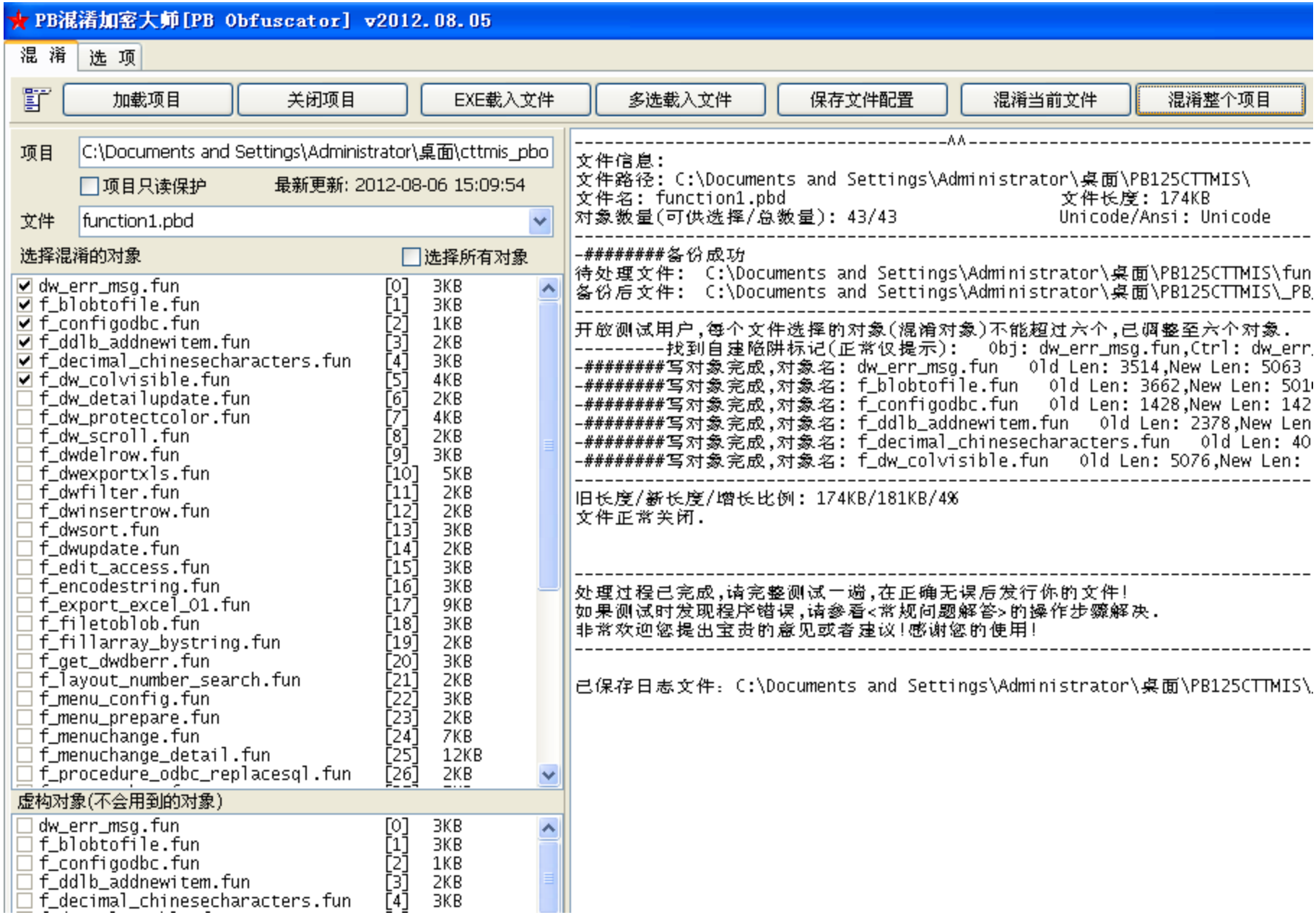
2012.08.06

目录

- 一.软件截图
- 二.基本原理
- 三.混淆原理
- 四.技术路线
- 五.软件特色
- 六.操作步骤
- 七.版本划分
- 八.如何购买
- 九.联系方式
- 十.开发日志
- 十一.常规问题

一.软件截图

返回目录


PB混淆加密大师[PB Obfuscator] v2012.08.05
混 淆
选 项
加载项目
关闭项目
EXE载入文件
多选载入文件
保存文件配置
混淆当前文件
混淆整个项目
项目
C:\Documents and Settings\Administrator\桌面\cttmis_pbo
项目只读保护
最新更新: 2012-08-06 15:09:54
文件
function1.pbd
选择混淆的对象
选择所有对象
dw_err_msg.fun [0] 3KB
f_blobtofile.fun [1] 3KB
f_configodbc.fun [2] 1KB
f_ddlb_addnewitem.fun [3] 2KB
f_decimal_chinesecharacters.fun [4] 3KB
f_dw_colvisible.fun [5] 4KB
f_dw_detailupdate.fun [6] 2KB
f_dw_protectcolor.fun [7] 4KB
f_dw_scroll.fun [8] 2KB
f_dwdelrow.fun [9] 3KB
f_dwexportxls.fun [10] 5KB
f_dwfilter.fun [11] 2KB
f_dwinsertrow.fun [12] 2KB
f_dwsort.fun [13] 3KB
f_dwupdate.fun [14] 2KB
f_edit_access.fun [15] 3KB
f_encodestring.fun [16] 3KB
f_export_excel_01.fun [17] 9KB
f_filetoblob.fun [18] 3KB
f_fillarray_bystring.fun [19] 2KB
f_get_dwdberr.fun [20] 3KB
f_layout_number_search.fun [21] 2KB
f_menu_config.fun [22] 3KB
f_menu_prepare.fun [23] 2KB
f_menuchange.fun [24] 7KB
f_menuchange_detail.fun [25] 12KB
f_procedure_odbc_replacesql.fun [26] 2KB
虚构对象(不会用到的对象)
dw_err_msg.fun [0] 3KB
f_blobtofile.fun [1] 3KB
f_configodbc.fun [2] 1KB
f_ddlb_addnewitem.fun [3] 2KB
f_decimal_chinesecharacters.fun [4] 3KB
文件信息:
文件路径: C:\Documents and Settings\Administrator\桌面\PB125CTTMIS\
文件名: function1.pbd 文件长度: 174KB
对象数量(可供选择/总数量): 43/43 Unicode/Ansi: Unicode
-########备份成功
待处理文件: C:\Documents and Settings\Administrator\桌面\PB125CTTMIS\fun
备份后文件: C:\Documents and Settings\Administrator\桌面\PB125CTTMIS_PB
开放测试用户,每个文件选择的对象(混淆对象)不能超过六个,已调整至六个对象.
---------找到自建陷阱标记(正常仅提示): Obj: dw_err_msg.fun,Ctrl: dw_err
-########写对象完成,对象名: dw_err_msg.fun Old Len: 3514,New Len: 5063
-########写对象完成,对象名: f_blobtofile.fun Old Len: 3662,New Len: 501
-########写对象完成,对象名: f_configodbc.fun Old Len: 1428,New Len: 142
-########写对象完成,对象名: f_ddlb_addnewitem.fun Old Len: 2378,New Len
-########写对象完成,对象名: f_decimal_chinesecharacters.fun Old Len: 40
-########写对象完成,对象名: f_dw_colvisible.fun Old Len: 5076,New Len:
旧长度/新长度/增长比例: 174KB/181KB/4%
文件正常关闭.
处理过程已完成,请完整测试一遍,在正确无误后发行你的文件!
如果测试时发现程序错误,请参看<常规问题解答>的操作步骤解决.
非常欢迎您提出宝贵的意见或者建议!感谢您的使用!
已保存日志文件: C:\Documents and Settings\Administrator\桌面\PB125CTTMIS\

二.基本原理

返回目录

什么是混淆器，什么是PB混淆器？为何要使用混淆器？

在编译型语言中，高级语言总被直接编译成汇编码，然后再到机器码，所以比较低阶，除了反汇编外，基本无法分析程序。而在PB和早期的VB，现在的Java，C#等都是相当高等级的带“动态特性”的语言，他们利用虚拟机技术或者层次更高的解释技术，将高级代码先编译成一种精简的中间执行码，然后被解释执行。与纯解释执行的vbscript等相比，少了重复n次的扫描和分析，所以认为比较快。

在PB中，比如：

```
long ll_temp
ll_temp = 100+100
```

这个最简单的算式，编译器先确定右边是两个字面量的数字类型相加(此处仅举例，也可能在编译时就事先加为200(看是否优化)，如果是ll_temp =100+变量，100就一定会编译到执行码里面)，在pb中默认的数字类型是long型，浮点小数是double型。当然也有其他的认定比如把10赋值给一个int型(16位)，则字面量可能被确定为int或者uint，这个是编译器自行判断，通常编译器都分得很细，针对每种语言界定的意义都有相应的表达方式。如上等式，编译后的形式大致为:

A．取值/long型/100;

B。取值/long型/100;

C。相加/long型加法；

D。数据类型转换(不是必须,但类型不同的数或者变量，必先做类型转换)；

E。赋值(将值或者地址/指针赋值给变量)

也许编译后的代码举例是：（HEX）

```
10 00 01 00          //取某个变量，代号为01
20 00                //取long型字面量（长度16位的码）
64 00 00 00          //字面量0x64，就是100(long占用4位，并存在本地，如其他类似C，C++中的指针类
                     //型，基于处理方便的原则，本地只保存4位指针(或叫地址),而实际内容保存在一个固定
                     //的段内。比如longlong类型(8位),或者浮点数，金融数，字符串，数组等都是长度不确
                     //定的，有个专门的结构来表达他们，而在代码内，只存放指针，操作时，因为知道数
                     //据类型，也就知道在何位置，如何存取相应的部分。
                     //所以取long的这种代码结构可以用于取任何数据。
```

基本原理 续

返回目录

```
20 00          //取第二个数字
64 00 00 00    //第二个100
50 00          //long型相加
5F 00 00 00    //赋值
```

这就是大概意义上的P-Code码。实际过程中，因为涉及到数据类型的转换或者提升，有许多的辅助代码。比如long和int相加，要提升int为long型后再相加。这是一个常识。当然还有其他与C，C++一致的要求，就是语法，比如优先级，这在P-code中也有实际的实现，否则就不成其为一种语言了。由于数据类型众多，而且PB对内嵌的SQL等支持很好，所以码表极多。如longlong类型和byte类型，try。。。catch都是在发展的过程中添加进来的。这样，到目前pb12.5为止，据统计，码表已经达到600多个。从PB5-12和PBK2.5总计码表6000余。每个码都各司其责，完成指定的具体任务。

不管是机器码还是P-code，都为执行码+数据，数据的长度由前面的执行码决定。有的没有数据，有的有多个byte的数据。如上第一行，01 00为变量的代号，占2个字节。因为一个字节(255个)不敷表示。

对于执行码与数据混合的这种形式，我们知道不执行前面的码，就无法预知后面某个位置的某个byte到底何用。也就是我们无法区分某个byte到底是执行码还是数据。一切都必须从第一个码开始执行，同样的，反编译是一个根据码表和数据反向解析和得到源码的程序，他是一个模拟的虚拟机，不过它执行的结果不是数据运算结果，而是得到我们的意图，我们程序员所写的源码。在机器码编译时，由于CPU只有在执行比较简单的任务才会得到较高的效率(加减，移位，异或，取反，判断溢出等)，CPU无法执行一个调用SQL的过程，但是在p-code中，调用SQL可以被编译成一个p-code码，然后再由虚拟机中的函数来具体处理，它的灵活性很高，能处理高阶的逻辑。所以硬件码都比较低层，而且近似数字处理的机械过程。而P-code不一样，它与源码有直接的对应关系。除了逻辑判断会有多种表达式用相同的实现而无法真正还原源码的写法外，就一个简单的a = b+c这样的算式来说，与源码是一一对应的。这也是现在PB反编译和Java，c#的代码可以反出来一摸一样，令我们瞠目结舌的缘故。我们知道这点对为什么要使用混淆器来保护我们的PB代码就有认知了。

基本原理 续

返回目录

混淆器发展了这么多年，有几种实现，如打乱顺序，然后再重新串联；抹掉一些遗留而本身运行不依赖的字符，如非公开的function/event名称，instance变量，本地变量都是不被外部访问的，所以编译器就在内部直接呼叫ID。而外部呼叫的全局函数，public的实例变量名，全局变量等，外部直呼其名。就不可以抹掉，只能全局替换(仅举例，有用无用要具体判断)。当然利用废指令，能扰乱正向分析而让反编译误入歧途，也是一个方法。

以现在的P-code编译或者所谓的字节码而言，最主要保证不被破解就是要守住反编译一关，否则编译后发行和发布源码无任何区别。需要反出源码的人当然最好是程序员，因为反编译后会有一些错误，程序员知道他自己的书写习惯容易恢复；而对于不想得到代码的人，他可以看到一些关键的数字，文字，逻辑判断的位置，而用Hex搜索，不下几分钟，就能定位从而修改Pcode。如把=改成<>，从前面的知识我们知道，只是一个byte的修改而已。如此简单的一个举例，只想说明一个问题，在这种P-code模式的编译情况下，反应的计算过程相当的高阶，也就是能对高级代码有一个直接的映射。这也就是PB等语言存在反编译器的缘故。而不曾听说编译型语言有反编译器一说。正是有如此特点，所以PB代码要保护起来有相当难度，但是我们知道，在很多论坛都对调试pbd有经验介绍，提得最多的是利用反编译器得到可阅读的源码再写注册机，当然这已经不是意义上的破解了。本质是反编译器在起作用。而亦有少数跟踪到vm内部去打转的说法，得出的结论就是调试很麻烦。所以不借助反编译器，是很难破解pb写的程序的。当然简单的程序，和碰到孜孜不倦的那种高手除外。因为堆栈，内存等都可以被调试看到，特别我们的注册过程又会生成如磁盘号，注册码等，如果跟踪还是容易得到的。不过，聪明的人都会在许多地方下“暗桩”,也就是叫破解者无所适从的方式。毕竟谁能把破解的东西用于生
产环境呢？用于生产还是存在风险的，暗桩就是一个不可见的黑洞。因为人们较高的研究热情，如.net和java都可以反编译得非常棒，当然pb也不例外。可想无混淆措施的pb代码等于源码。

我对以上的内容了解有限，在pbd格式分析过程中也只领悟到这些。并且我也不是调试能手，所以就此打住。我们的任务就仅限于阻止反编译器。混淆器就是通过改变代码存放的物理顺序并重新串联，以不改变其执行逻辑，而阻止其顺序分析。还有如去除部分文字，函数，事件名，参数名字，再加上特意制作的虚假对象，虚假函数事件，或者虚假的代码片段等等措施来达到我们保护的目的。

三.混淆原理

返回目录

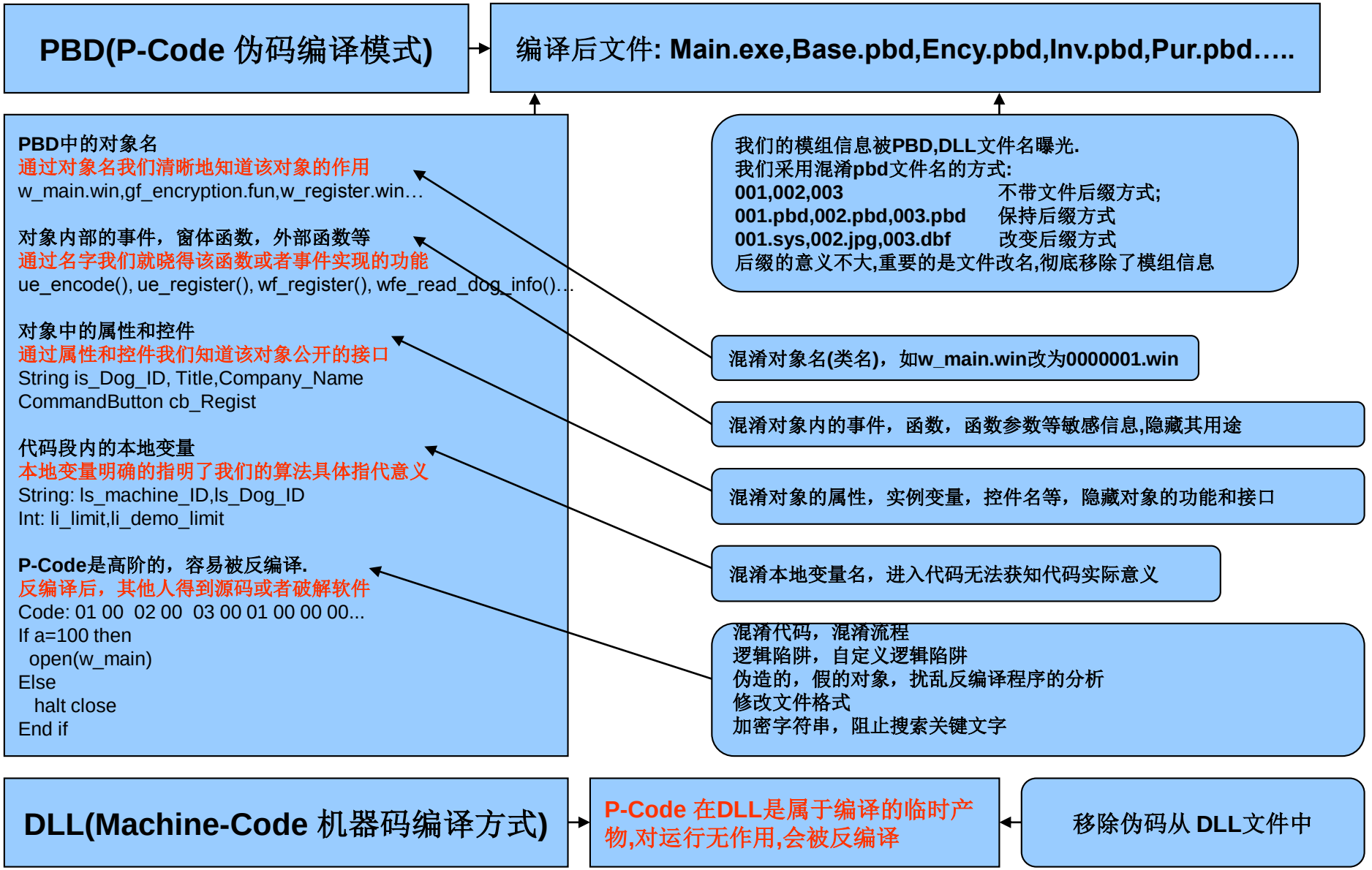
本软件防止反编译措施:

- 代码拆分并任意位置摆放再串接---阻止顺序分析
- 虚假和故意的跳转
- 破坏变量名和函数参数,函数名---降低可读性
- 程序员自定义逻辑分支陷阱—部分二进制会被破坏
- 虚假废对象--二进制会被破坏
- 虚假废函数和事件--二进制会被破坏
- 混淆对象名[暂未发行]
- 明码字符串处理[暂未发行]
- 修改文件名—降低文件名可读性，降低模块识别
- 保持离散性，不同电脑，不同年月，不同的版本算法有所不同，无法测试到所有的算法
- 混淆过的文件无法看出使用的混淆器的版本，不同的版本之间存在较大的差异

四.PB混淆器技术路线与设计实现

返回目录

2009.07---2011-05-08
http://www.mis2erp.com



四.PB Obfuscator Technology Roadmap

返回目录

2009.07---2011-05-01
http://www.mis2erp.com

PBD(P-Code Compiled Mode) → Main.exe,Base.pbd,Ency.pbd,Inv.pbd,Pur.pbd…..

Object within the pbd
We know that the use of the object,by Object's Name
w_main.win,gf_encryption.fun,w_register.win…

Event/Function/External-Function within the Object
We know that the use of the Enent/Function,
by Event/ function's Name
ue_encode(), ue_register(), wf_register(), wfe_read_dog_info()…

Property and Control within the Object
We know that the use of the Property or Controls
String is_Dog_ID, Title,Company_Name
CommandButton cb_Regist

Local variables within Code Segment.
We know that the use of the veriable
String: ls_machine_ID,ls_Dog_ID
Int: li_limit,li_demo_limit

The P-Code Can be Reversed,Turn to Script.
We know that the use of the Code
Code: 01 00 02 00 03 00 01 00 00 00...

```
If a=100 then
  open(w_main)
Else
   halt close
End if
```

Module information, Be displayed by the PBD-file name
Important objects will be found in the pbd
Confuse the PBD File Name as follows:

001,002,003	Without Extension;
001.pbd,002.pbd,003.pbd	Does Not Change Extension
001.sys,002.jpg,003.dbf	Change the Extension

Hide the module information

Confuse the Object's Name In PBD File

Confuse Event/Function/External Funciton's Name,and Parameters

Confuse Instance Variables/Properties and Control's Name

Confuse/Remove Local variable's Name

Confuse Code Segment, Change process
Logic Trap, Logic Trap Customized
Fake object,Fake Funciton and Event
Encrypted string: To prevent the search of sensitive information

DLL(Machine-Code Compiled Mode) → P-Code Segment is Remnant, Can be reversed ← **Remove the P-Code from DLL**

五.软件特色

返回目录

- 以往的混淆器

主要功能：仅仅打乱代码摆放位置

缺点: 使用智能匹配算法可以逆向,使用大量调整算法可以还原代码顺序

- 该软件

1. 增加多种算法,构建逻辑分支,使反编译不容易逆向
2. 允许程序员参与构建一些奇异的分支
3. 清除变量,函数参数,函数名等可读文字,使反编译绝对无法得到可以重新使用的源码(消除95%以上的反编译企图)
4. 清除文件格式标记
5. 设置虚构的对象和函数使得反编译无法判断真假
6. 其他一些措施随升级后加，某些措施不会写入文档和明示。

六.操作步骤

返回目录

登录和认证过程，大约需要1-2分钟，认证后可以离线使用

//向多个主机发送认证数据
上传凭证成功,在主机: 1,尝试: 1
上传凭证成功,在主机: 2,尝试: 1
上传凭证成功,在主机: 3,尝试: 1
上传凭证时,成功的主机数: 3
请等待30-60秒,以接收服务器的认证结果...

//下载到返回结果
下载凭证成功,在主机: 3,尝试: 1

//显示授权内容
//测试用户
现在是开放测试时间,请开始测试吧...
提供测试的版本有: PB5.0,6.5,7.0,8.0,9.0,10.0,10.5,11.0,11.5,12.0,12.5
测试版仅供评估(有弹出提示窗和运行时间限制),请勿用于正式发行文件的处理.

服务器: China.guangdong.dg

通过在线验证,您将使用正式版.....

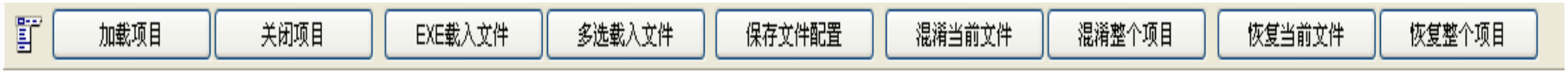
//显示授权内容
//正式版用户
机器码: 允许　文件校验: 允许　用户级别: 专业版　授权期限: 2020-12-31 允许
授权版本: 60/70/80/90/100/105/110/115/120/125
服务器: China.guangzhou.serv01

通过在线验证,您将使用正式版.....

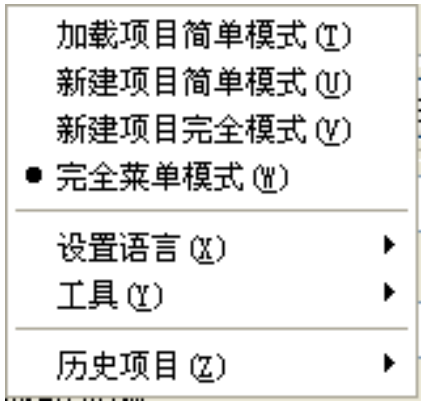
如果存在无法连接到服务器的情况，见这里说明。

6.1菜单与按钮

返回目录



菜单切换,为满足不同用户的需求,左上方有一个弹出菜单,单击可以进行按钮的切换



四种模式保持操作的简洁性
配置完项目后，你可以切换到“加载项目简单模式”会只留下两个按钮
加载过的项目，会出现在历史项目中，便于快速开启
菜单还包括：语言设置,网络检测工具

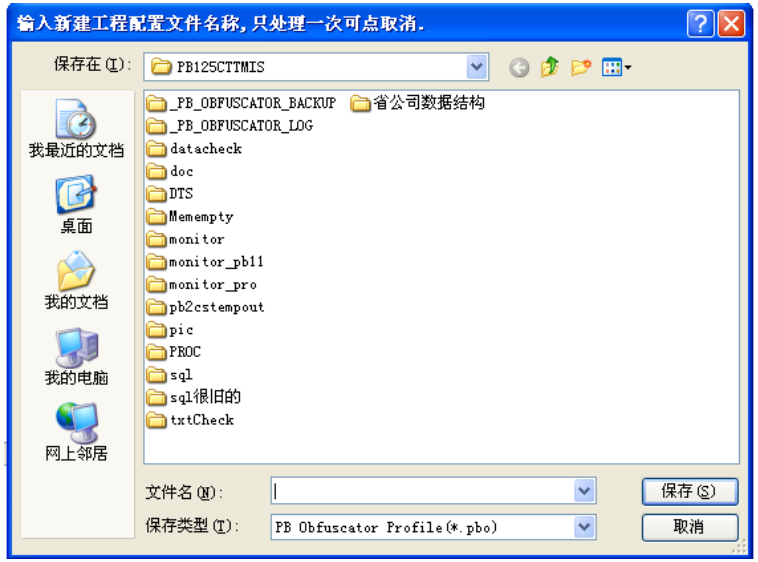
操作步骤-新建项目

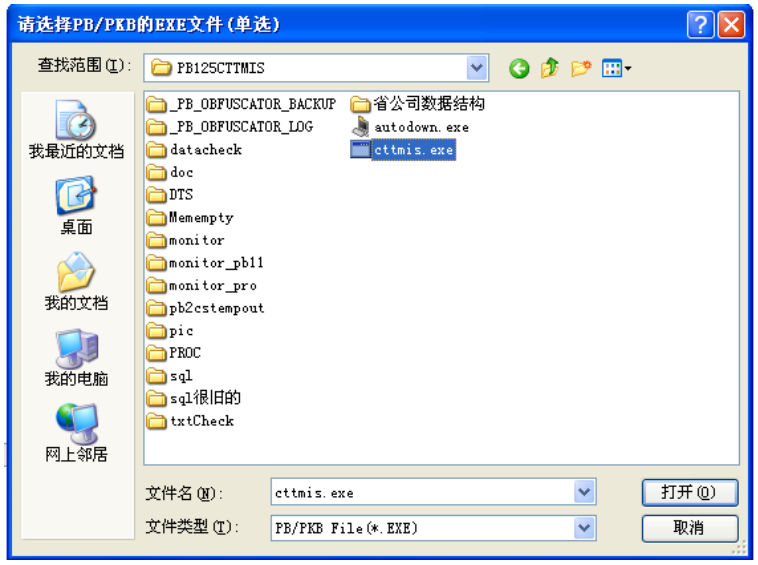
返回目录

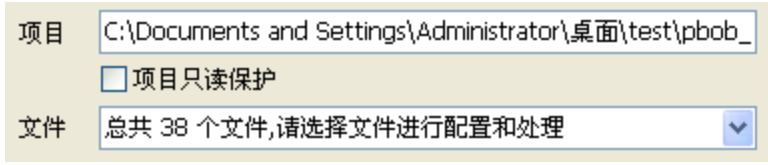
6.2 新建项目-从exe载入文件列表

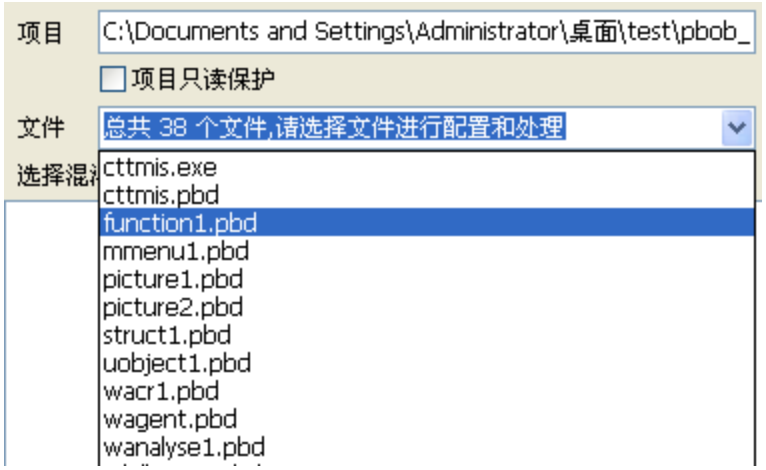
EXE载入文件

6.2.1 点击该按钮，按照提示输入工程名，并选择程序的主exe文件



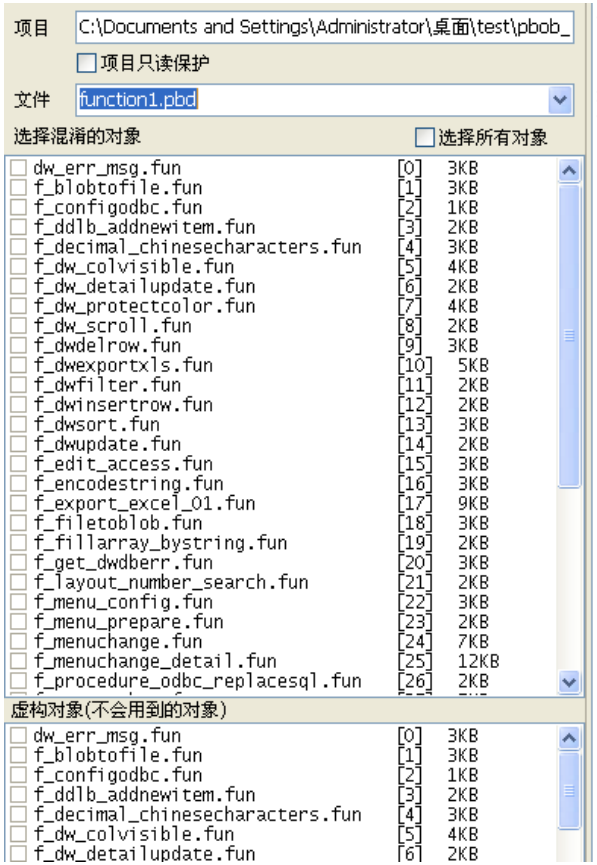


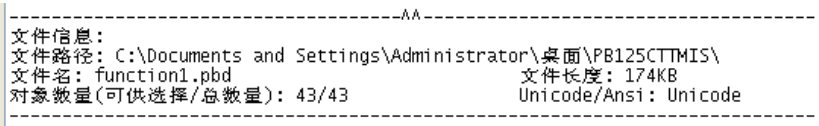




6.2.2. 从下拉列表中选择某个需要混淆的文件
6.2.3. 勾选需要混淆的对象(上)，也可以勾选虚构的对象(下)

返回目录





6.2.4. 切换到“选项”页,配置项目选项和文件选项,注意每个文件可以保存单独的配置

返回目录

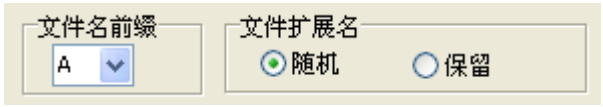
返回目录

6.2.5 具体配置选项的说明会在后面给出详细解释,配置完一个文件后,点”保存文件配置“按钮.项目选项会在每个文件配置保存时都保存一次,所以你可以在配置任何文件时配置项目的选项. 保存某个文件的配置后,文件下拉框中,该文件名会带一个星号.表明它被配置过。下面我们逐一解释选项配置参数：

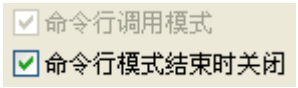
将可读的对象名修改为不可读的字符

返回目录

将可读的文件名修改为无意义的字符



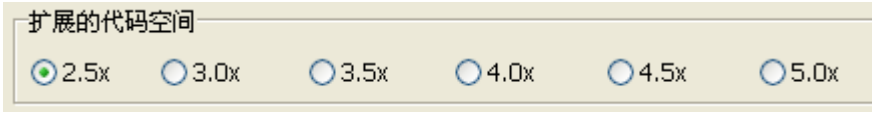
前缀用于区别不同程序所属的文件,如有两个程序都放在同一个目录，分别叫A.exe和B.exe，那他们的pbd就可以这样区分。
A001.pbd，A002.pbd。。。
B001.pbd，B002.pbd。。。
文件扩展名如果勾选“随机”，后缀名将被修改，改成.jpg，.bmp等



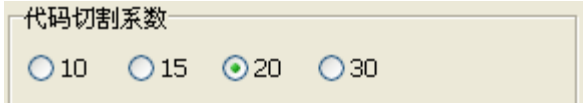
命令行调用模式如果被授权使用，你可以使用程序名+参数(工程文件名)来运行该程序并在结束后退出该程序，关于命令行模式的详细情况见这。

处理过程将被保存下来，位置：
PBD所在文件夹下面的_PB_OBFUSCATOR_LOG目录中

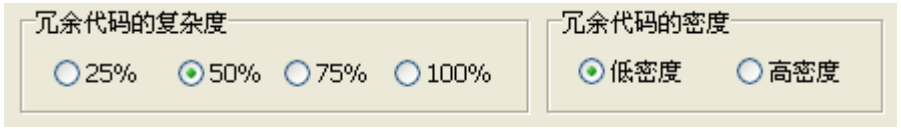
默认情况下(在有授权的情况下),软件会清除一些可读的文字，比如本地变量名，以使反编译出的代码失去价值。在将PBD给第三方去开发和扩展时，要去掉该勾选，以保持第三方调用的成功。关于混淆函数名见这里。



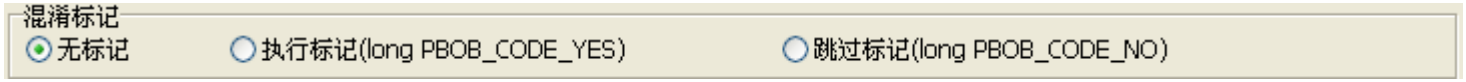
为了放置一些分支和冗余代码，所以必须扩展代码空间。如果出现“无法找到足够的空间放置Code-Chip”提示时，请适当调高此参数



切割系数指：多少个码切分一次，数值越小，表明切分次数越多，相应的需要的扩展的代码空间也越大。对代码特别少或者特别多的代码段，程序内部会稍做调整。这个系数也会被动态改变。不会是固定码数的切分。



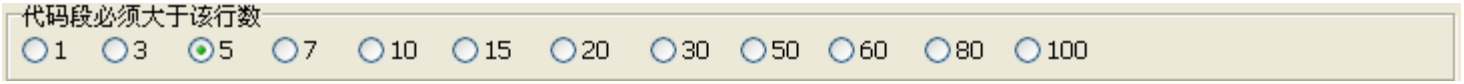
复杂度和密度都如字面意思所表述的那样 *返回目录*



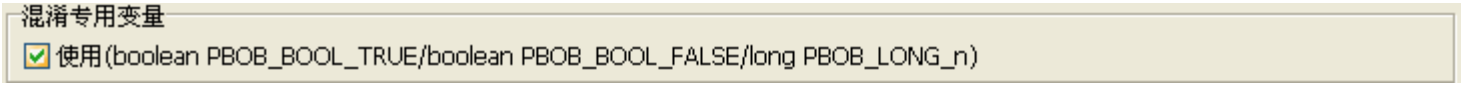
混淆标记代表三种情况：

1. 无标记，那全部代码都要混淆处理
2. 执行标记：默认都不进行混淆，除非你在本地变量处声明了变量：long PBOB_CODE_YES
3. 跳过标记：默认都混淆，但是允许跳过而忽略，如果声明变量：long PBOB_CODE_NO

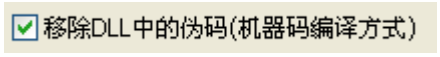
这三种方式用于满足一些复杂的工程需求，特殊的情况



为了提高处理后程序的执行效率，故考虑设一个最低行数的参数，这样某些一两行，三五行代码的代码段就会跳过而不处理，从而提高运行效率。默认值是5行，满足较多的情况。你也可以根据实际的情况做调整，但是记住不要让自己真正要保护的关键部分的代码因为这个条件被跳过。



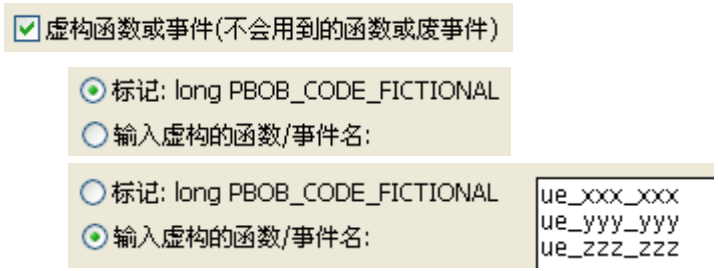
混淆专用变量会被使用到内部的混淆和分支算法上,请在本地变量处声明它，你必须闲置它们(仅声明不使用)。[如何声明它们见此](#)。



移除DLL中的伪码残留部分

Create 和Destroy事件都是控件的创建和销毁，没有加密的价值

自定义逻辑陷阱是一块逻辑上执行不到的分支块。它将被从二进制上去破坏掉，目的是阻止反编译。我们知道固定模式的特征很容易捕获和自动处理，而逻辑意义上的分支很难由程序去判断。特别是带有运行时才决定的结果值，是很难由反编译器静态分析得到的。
关于自定义陷阱见这里详细解释。

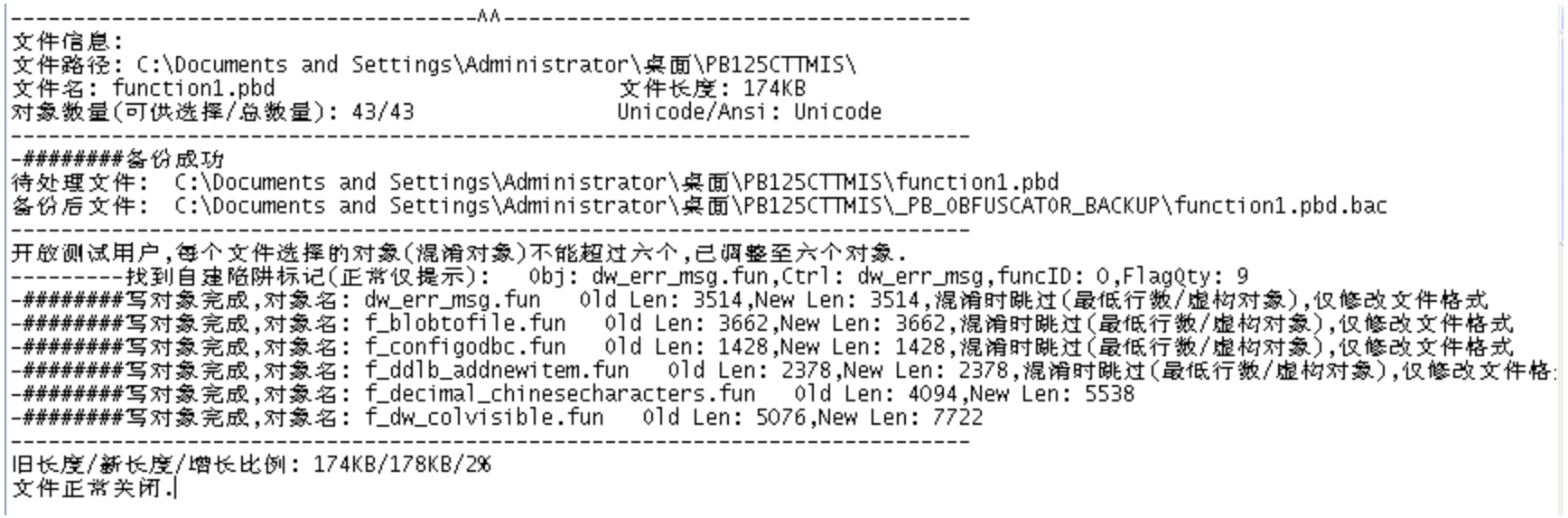


你可以从正常的函数复制(另存为)一份重命名然后作为虚构函数,你可以在虚构函数的Local variables处声明一个标记变量：
long PBOB_CODE_FICTIONAL
注意: 声明即可无需初始值
关于这个变量请看这里.
也可以输入虚构函数的名称，如果你定义一个命名规则，那可能你的所有虚构函数都是有相同或相近的命名,就可以在这里输入，这样比写标记变量要方便。

返回目录

混淆整个项目

6.2.6 配置完需要的文件后,点”混淆整个项目“按钮，一键完成项目的混淆
如果勾选了保存日志,处理日志将会被保存到:
PBD所在文件夹下面的_PB_OBFUSCATOR_LOG目录中
备份文件在：
PBD所在文件夹下面的_PB_OBFUSCATOR_BACKUP文件夹中
注意这两个文件夹中的文件不要被发行！



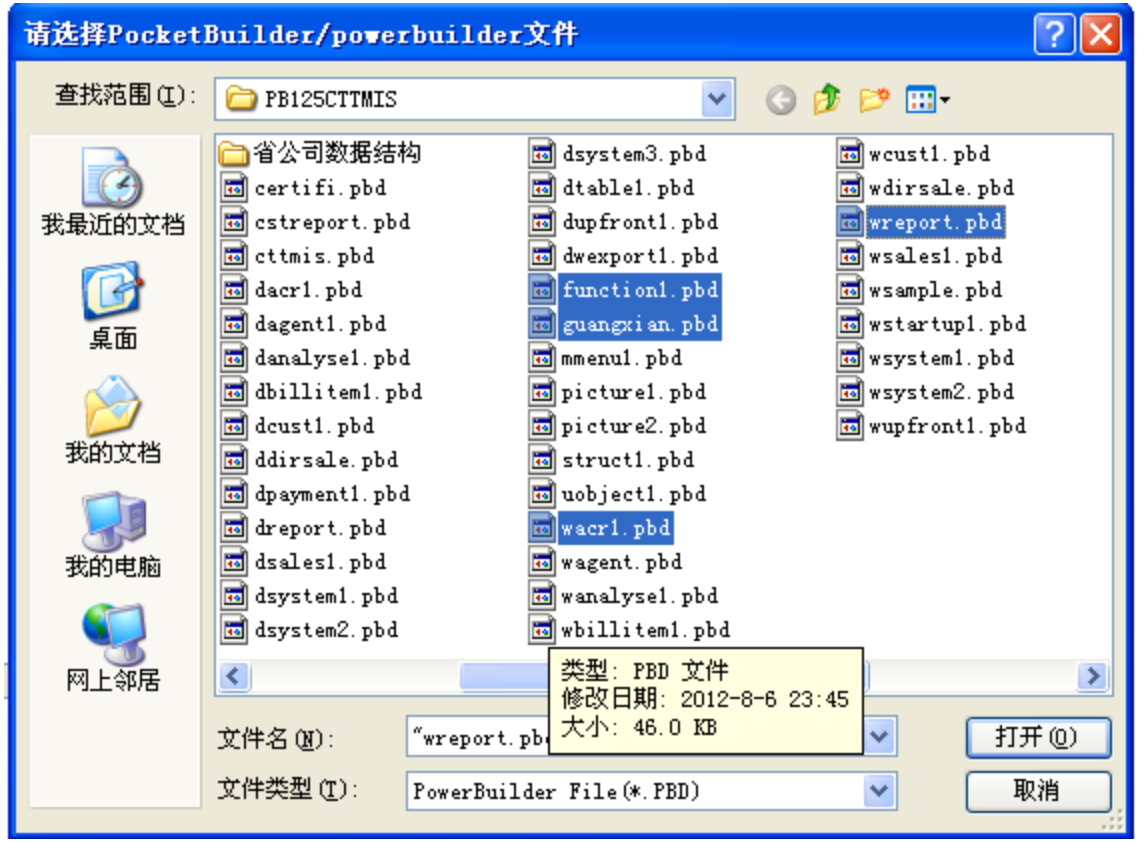
返回目录

6.3 多选载入文件

返回目录

多选载入文件

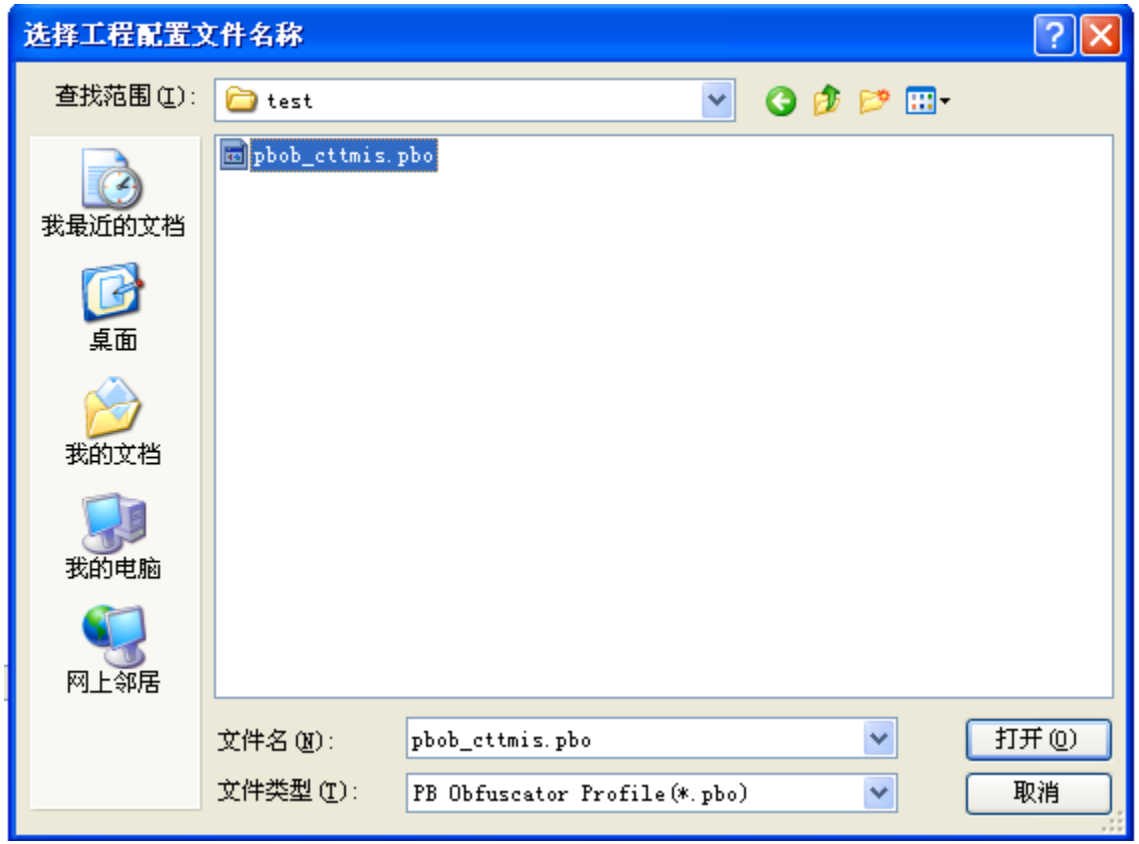
6.3.1 多选载入适合处理少数文件或者文件需要动态加载的情况
(文件不全在Exe中的library List中,所以通过手工选择载入)
载入后配置和执行混淆的过程同6.2的步骤

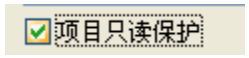


6.4 加载项目

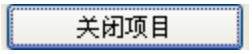
返回目录

6.4.1 加载已经存在的项目时,点击此按钮,选择工程名
打开工程后，可以从文件列表中选择和切换文件，并查看文件的配置情况
其执行混淆也是点击“混淆整个项目”





6.4.2 如果此处打钩,项目中的文件配置和项目配置不允许再改变
配置完成并混淆无误后，建议打钩保护起来



6.4.3 如果操作一个项目后，要新建其他项目，请点”关闭项目”

6.5 工程变更后的更新

返回目录

6.5.1 当exe的PBD并没有发生变化，而只是某些PBL中的对象发生了变化，比如增加删除了对象时，请加载工程后，在下拉中选择对应的文件，选择文件时，文件中的对象名就会重新刷新(读取最新的对象列表),你可以根据实际情况修正，然后保存即可。其他混淆操作等相同。

6.5.2 如果exe中的PBD发生了变化，比如增加或者减少了PBL，则先打开工程，然后点“exe载入文件“，将从exe中重新读取库列表，如果某个pbl未变动，当你选择下列列表时，先前配置好的参数会带出来，如果是选择新建立的文件，则打钩和配置完，点保存即可。

6.6 工程的覆盖

6.6.1 如果在新建一个项目时，输入的工程名已经存在，会给出提示，如果选择覆盖，已经存在的工程文件会被覆盖重写。

转到第七节

自定义逻辑陷阱

返回前面

自定义逻辑陷阱的标记为：

```
Long ll_mylong                //声明变量
ll_mylong = 123456789+123456789+123456789+123456789+123456789+123456789        //”自建陷阱”的标记
```

记得表达式是: 等号左边为一个局部变量，long型，变量名任意而无需固定，等号右边是6个“123456789“相加。混淆器在遇到这个算式时，将把代码变成任意跳转或者垃圾代码，或者伪造的执行码。
当然这个算式要放在“永远无法执行得到“的地方，比如：

```
Choose case  ll_mylong
    Case n1                          //执行不到的case
        //建议加一个messagebox提示,万一写错逻辑可以提示到
        //messagebox("提示","混淆逻辑时写错")
        ll_mylong = 123456789+123456789+123456789+123456789+123456789+123456789
    Case n2                          //执行不到的case
        //建议加一个messagebox提示,万一写错逻辑可以提示到
        //messagebox("提示","混淆逻辑时写错")
        ll_mylong = 123456789+123456789+123456789+123456789+123456789+123456789    //可以多行
        ll_mylong = 123456789+123456789+123456789+123456789+123456789+123456789    //可以多行
     case 。。。
        。。。
End choose
```

提示：
建议加一个messagebox到正常逻辑执行不到的地方, 万一写错逻辑可以在PB中运行时提示到。

```
messagebox(“提示”,“混淆器专用区域不能运行到这里!")
```

自定义逻辑陷阱

返回前面

当然还可以其他的方式，如：

```
If myfunc1()=myfunc2() then            //在反编译时，是很难判断你这个表达式的结果的
    ll_mylong = 123456789+123456789+123456789+123456789+123456789+123456789
End if
```

原则：尽可能利用复杂表达式，本地变量，全局变量，函数等来写条件，就如你平时写程序一样，这样不容易被机械地逆向。只
是需要这样的条件恒false，即永远不要执行到这个标记所在的行。
比如：

```
//这个条件,反编译很难去判断结果。因为是程序员写的，无固定模式。
If gf_fun1(p1,p2,p3)+100< gl_temp+ll_temp then
    ll_mylong = 123456789+123456789+123456789+123456789+123456789+123456789
End if
```

当然也可以在for循环中插入一个continue，然后写上这个标记。也可以在return之后写上无数个这个标记。也可以用goto语句跳过三五行这个标记(goto跳过肯定不执行)。只要满足“永远执行不到”就行。

现在内置的一些算法模式相对还是比较呆板。注意，为了保险起见，关键代码内建议至少三五个这种标记。当然你写几十个也可以。同时，可以在执行不到的地方编写自己的冗余代码并用goto跳转到不同的行，反编译的结果中如果含有较多的goto和label，会被认为是不可信的代码。在执行不到的地方写goto当然不在乎它具体跳到哪里去。

虚构函数的标记

返回前面

long PBOB_CODE_FICTIONAL

在函数内部的本地变量处声明了此变量,该函数的数据将被涂改,这种涂改是随机的.

声明有该变量的函数,应该是不会在任何地方使用到的,你可以从某个函数复制来生成这种函数,并有合适的命名规则以便区分,不要把正常要使用到的函数中声明此变量.

提示:

建议加一个messagebox到正常逻辑执行不到的地方, 万一写错逻辑可以在PB中运行时提示到。

messagebox(“提示”,“混淆器专用区域不能运行到这里!")

混淆函数名的标记

返回前面

- long PBOB_FUNNAME_YES

函数/事件的本地变量处声明此标记,函数名字将被混淆。你必须保证不在该对象内部和其他代码中使用”函数名”的字符串方式调用，以及不使用动态Call的模式来调用该函数,假设w_main中有函数uf_xxx,事件ue_xxx，则情况如下:

```
w_main.uf_xxx(0)                    //正常能执行
w_main.dynamic uf_xxx(0)            //报错触发systemerror
w_main.dynamic event ue_xxx()       //执行不到ue_xxx这个事
                                    //件,也无任何错误提示
w_main.triggerevent(“ue_xxx”)   //执行不到ue_xxx这个事件,
                                    //无任何错误提示
w_main.trigger event ue_xxx()   //可以执行到
```

*全局函数不用此功能，因为对于全局函数而言，对象名未混淆的话，对象名和函数名是相同的。如果后续升级实现[对象改名]时可以达到此目的。

混淆专用变量

返回前面

- boolean PBOB_BOOL_TRUE = true
- boolean PBOB_BOOL_FALSE = false

以上两个变量名固定，初始值固定。

- long PBOB_LONG_n = n

以上Long型变量请保持左边的n和右边的n
相等，n为正值。例如：
long PBOB_LONG_1000 = 1000
long PBOB_LONG_10000 = 10000
你可以声明多个这样的Long型变量,比如三五个.

请仅仅声明和初始化它们，而不要使用它们，内部算法将随机改变它们的值并用于专用目的。这些变量可以不用放置在Local变量的最开始，可以任意，但是在第一行代码之前。
记得它们必须声明在Local variables处。

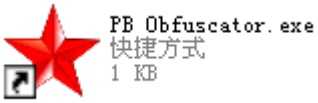
命令行方式调用

返回前面

1.可以建立bat文件来执行混淆, bat中如下书写，参数不要用引号

```
@echo off
echo PB混淆加密大师 命令行调用模式
echo PB Obfuscator CommandLine mode
"F:\pro\PB Obfuscator v2012.08.07\PB Obfuscator.exe" C:\Documents and Settings\Administrator\桌面\test\pbob_cttmis.pbo
exit
```

2. 可以在其他程序中直接调用
因为存在反调试代码，所以你还是必须建立bat。然后用cmd ***.bat这样的方式运行。



3. 建立快捷方式

3.1可以在PB Obfuscator.exe上点右键-》发送到桌面快捷方式，并在快捷方式的属性-》目标 处增加工程文件的路径作为参数：
"F:\pro\PB Obfuscator v2012.08.06\PB Obfuscator.exe“ d:\test\test.pbo
参数可以含有空格，长文件名目录等，但不要引号。程序从第一个空格处切分并把后面当做工程文件的全路径。记住是全路径。

3.2 可以在PB Obfuscator.exe上按下右键拖动-》右键菜单-》在当前位置建立快捷方式，然后添加参数，步骤同上。

以上三种方式，都会运行界面，并通过认证，然后自动载入工程文件，并混淆整个工程。根据配置，也可以在处理完成后自动退出程序。

关于无法访问服务器完成认证

返回前面

- 左上角菜单中有网络检测工具可以做PING和DNS解析检测
- 国内DNS服务器有存在过滤问题，有时无法解析我们的网站，可以采用这个DNS：

  168.95.1.1
- 请注意上传凭证和下载认证结果的提示，以便报障

关于下划线开头的对象名无法被混淆

返回前面

在解析pbd格式时，要排除一些系统内定的对象如“_initsrc”
这个基本是每个对象都会带有。
还有几十个属性或者函数如:
__destroy_object，__set_attribute，__set_attribute_item，__get_attribute等。
因为无法知道到底它们一共有多少以及是具体是哪些，每个版本又互不相同，所以只有抓第一个字符为下划线的对象或函数做特别处理。
所以建议不要采用下划线开头来命名，包括对象名，函数和事件名，以及属性名，变量名。如果要特别标示，请改用我们pb规则中不用的前缀。当然不用混淆的话不存在此问题。

现在程序已经修改为提示并略过下划线开头的对象。注意看混淆后产生的提示信息。记得下划线开头的对象被忽略而不处理。

七.版本划分

返回目录

正式版分三个用户级别:

模块	说明	专业版	企业版	高级企业版	测试版
工程配置文件	一次配置,多次运行,一键混淆	*	*	*	*
命令行调用执行	批处理或者其他程序调用自动完成	-	-	*	*
代码混淆	切分代码,重新摆放,再次连接	*	*	*	*
内置混淆算法	内置的混淆算法	*	*	*	*
虚构对象	复制或虚设的对象,并涂改字节	*	*	*	*
虚构函数	复制或虚设的函数或,并涂改字节	-	*	*	*
自定义逻辑陷阱	自定义的一些字节,并涂改字节	*	*	*	*
移除DLL伪码	清除DLL编译时留下的伪码	-	-	*	*
修改PBD文件格式	把一些格式化的标记去掉	*	*	*	*
移除变量名	把明显的可读文字移除,消除可读性	*	*	*	*
移除函数参数	把明显的可读文字移除,消除可读性	-	*	*	*
移除函数名	把明显的可读文字移除,消除可读性	-	-	*	*
混淆对象名[暂无]	把对象的名字混淆,消除可读性	-	-	*	*
混淆文件名	把文件名混淆,消除模组信息	-	*	*	*

上表中 - 无此功能 * 含有该功能
不同的版本在强度，复杂性方面有些差异
不同的电脑，不同的日期，内部算法存在差异性
各个版本在服务及时性和技术支持方面略有差异

八.如何购买

返回目录

- 请来邮件或者加QQ,MSN索取报价
- 专业版可分三次分期付款
- 企业版可分三次分期付款
- 高级企业版可分六次分期付款
- 可以通过淘宝,支付宝,银行转账,银行异地存款四种途径支付
- 国外用户仅支持西联汇款方式
- 考虑这样的分期付款和付款途径，只为降低用户的支付压力，先购先用，并降低购买方风险。

八.联系方式

返回目录

- 网址: http://www.mis2erp.com
- MAIL: chengang0769@gmail.com
  chengang0769@21cn.com
- MSN: chengang0769@live.cn
- QQ: 27-3939-617
- 手机: 137-9029-9411

国内联络主要通过QQ和MAIL,比如售前询价和付款后联系,技术支持等.技术支持依照中国北京时间10:00~20:00.

国外联络主要通过MSN和MAIL.

通讯语言请选择中文简繁体或英文.

十.开发日志

返回目录

2012.08.06 发布2012新版，支持工程文件配置，一键完成混淆，也支持命令行调用
支持文件改名
内置算法的重大调整，改写算法并加强和变异。增加混淆专用变量和其他的一些标记
增加12.5的支持
修正pb5码长的问题

2011.10.23 重新加壳发布v2010.11.01新的build。

2011.04.26 发布v2010.11.01 修正后的build

2010.10.19 发布稳定的v2010.11.01版本

2010.09.22 (阴历中秋) 测试非简体语言，英文界面报时间格式错误strtodate not a valid date format，修改bug；另发现一个rev-szchar缓冲区错误，修改bug。

2010.09.10 增加用户可以自定义的“逻辑陷阱”，弥补靠内部算法无法构建复杂的此类陷阱的不足。并修改成
正式版，在线认证。修改界面支持中英文自动切换。
版本改为v2010.10.01

2010.07.02 网友PB8告知，他用到几个以下划线开头的对象无法混淆，已回复他：本软件不支持下划线开头的对象名，下划线保留给了“判别PB内置对象”用。*见此解释*

2010.06.18: 接网友PB8的报告，在20x处理时出现一个低阶的bug(空格字符串出现乱码错误)，修正。版本改为：V2010.06.1;

2010.05.27: PCM报告一个UO引用错误，经查为普遍错误。PropertyClear error(0-08)。修正。可能这个错误会影响到相当多对象，故提升版本为v2010.05.4

2010.05.19: 增补与整理pb5，10.5的码表，并测试5，10，10.5的pb-sample通过；
自此，支持5-12全系列。因pb12刚出正式版，前期只测试过beta，待后期着重测试。版本改为v2010.05.3。
NOD一个bug修正.

2010.05.17: 修正在checklistbox中未列出的对象如struct，menu等的“混淆标志”未清零的bug,致使未列出对象
的参数被非法修改，致使运行错误。bug由pcm和群内的牛解庖丁(475392*)测试时发现的,原因是初始化问题。限制用户必须勾选至少一个对象，否则无法得到某些参数的正确的初始化值。
版本更新到2010.05.2

开发日志 续

返回目录

2010.05.15: 修正短代码扩展系数太大造成的越界问题

2010.05.12: 修正TRL和FRE造成的问题。并测试自己的两个pb11编译的项目，运行正常。
修改版本号为2010.05.1

2010.05.11: 增加移除dll机器码编译时的伪码，一个网友说可以抹掉。但是我无法确定，故为一个选项供用户勾选。但是并未移除，而是先混淆，然后将入口地址拿掉，成无头僵尸，起到相同作用，因为我不想为此改动太多代码造成新问题。

2010.05.07: 修正try...catch结构的转移地址

2010.05.02: 增补pb10.5增加的byte类型的码约十个

2010.05.01: 对pcm的样本测试，发现10Bx标志处最后一个short不能为0，因为采用随机数扰乱，故测试大约十多次才发现一次，修正。

2010.05.01: 经pcm提醒，当我用自己的pb9测试时，6x标志可抹掉仍然运行正常，而他寄来的样本(也是9)就是不能抹掉，看提示好像是n_cst_msg，好像是PFC的东东，此处只好不抹。

2010.04.30: 反编译研究发现全局函数重载(内含一个以上的函数体造成报错)修正。

2010.04.29：与反编译器一起修正6，7，8与9严重不同：取ctrl-list的不同，从而6-12测试解析正常了。

2010.04.29：与反编译器一起修正：取控件名时判断控件的编号时 >=8000 写成了 ==8000,导致严重遗漏从而解析出错。修正。

2010.04.28: 对超长代码段实行折中妥协，自动降低JCP复杂度使得空间不够的提示尽量不出现。当无法插入JCP时，使用简单JXP做连接还原代码执行顺序。
对超短代码段提升长度为200字节。

2010.04.23：public的function和event需要用名字呼叫，故不抹掉。

2010.04.21: 修改exe格式丢失结尾的TRL段问题

2010.04 受网友建议，因pb11有代替9的趋势，感觉10以上混淆器是空缺并迫切需要，遂赶工完成并发布PB混淆器beta版。编程耗时40天。

2010.03.27 Foxstudio在2010.3.27寄了一份pb5开发工具绿色版，从而有幸使得PB版本近乎完整。Pkb未下载到2.0，只研究了2.5beta。发觉借用pb9的技术，故2.0无异。

2010.03 认真分析Powershield混淆方法和模式，写出反混淆单元，并以一套国产POS为例，反编译并修改字节获得无限试用期。总结其方法和缺点，验证了其模式化造成容易被“反混淆”的不足。为后续混淆器开发奠定了基础。如仅为其升级版，简单复制它的方法，无任何创新，不足以保护pbd文件不被破解。

开发日志 续

2009.08-2010-02 开发反编译器部分代码至90%，因其格式艰深，无任何文字可参考，多个关口差点”放弃”。且要兼容11个版本，屡屡废弃代码重写，前后耗时6月余，年后由于其他项目拖累，尚未释出。

2009.07-2009.09 分析pbd文件格式，可见我blog上零碎文字。

感谢的话：

感谢各位网友的鼎立支持！特别感谢发来邮件或者csdn-blog留言，群组提交意见的朋友们！感谢敢于测试和使用的勇士们！在开发日志的结尾不具名一并致谢！

九.常规问题

返回目录

除了这个软件之外，还有什么方式保护PB代码？

国外有PBProtect，但是是针对源码混淆的，有些特性不支持，收费也比较贵。

早期有Powershield，针对PB9以下，现在可以被逆向。

PBE，移除DLL伪码，仅仅支持PB9以下。

该软件处理速度如何，比如我的项目有15~30个pbd这样的规模

现在新版的速度已经非常快，大概1-10秒钟处理完一个文件。15-30个这样的文件，如果里面全部是dw的不需要配置和加密。整个处理过程远比你在PB中编译一个项目要短得多；对于经常升级软件(定制程度高和维护老系统)的用户，是非常方便的，不会耽误太多时间来做这个混淆步骤。

认证方式是什么，需要网络怎么设置？

认证方式是在线网络认证需要防火墙和路由器允许本机访问远程服务器的21端口。同时本地的DNS要能解析pbde.mis2erp.com/cnhost.mis2erp.com/ushost.mis2erp.com等域名.

认证后的提示是什么？

主要提示验证是否通过，开通的版本，授权的时间期限等

认证可以用多久？还是说每处理一个文件都需要在线运算？

认证完后可以离线使用，直到软件提示你需要重新开启为止。

正式版处理的文件，是否会过期？

正式版处理过的文件不会过期，也不存在弹出窗口。

如何优化和兼顾不同的对象特点来设置参数，以达到最佳状态？

常见对象分为几种：

1. 普通的用户交互界面，数据库不外乎增删改读。如果程序不敏感，不建议做混淆。

常规问题 续

返回目录

2. 授权部分，加密狗，网络认证，ftp认证，http认证，上传下载，机密数据，机密算法，这些建议混淆加密。而且必须用“混淆专用变量”和“用户自定义陷阱”，“虚构对象，虚构函数事件等措施来加强保护，这部分对象是保护的关键。
3. 用户自定义组件UO。如果不敏感，不建议加密，如果特殊独特，建议加密。
4. 常规组件，比如导出excel等组件，大家都有，不建议加密。
5. 大循环内的执行函数，事件等，不建议加密，因为假设混淆后增加执行时间5%，则根据循环次数将会把增加的执行时间放大N倍，这个视自己的需要而定。
6. GUI绘制组件，因为需要较高的执行效率，不建议混淆。
7. 高速算法组件，比如MD5组件，需要较高执行效率，不建议混淆。

连接无回应如何处理？

首先使用左上角的弹出菜单中的网络检测工具，检测PING和DNS，如果无问题，再检测自己的Router是否开放必要的端口，如果无法解决，请报故障，并提供详细的故障现象描述。

操作的简单步骤是什么？

第一次操作，先新建工程，一般从exe载入，如果PBD有动态添加的情况，请”多选文
件载入“，然后仔细对每个文件进行配置，配置完，以后执行混淆只需要加载项目并按”混淆整个项目“，混淆完检查混淆日志，并完整测试一遍无错即可发行。

几种编译方式的保护措施？

PBD方式编译，请按本文档的操作，DLL编译方式，必须勾选”移除DLL中的伪码“，其他措施和方法相同。

返回目录

如何测试混淆后的程序？

在你决定使用该产品保护自己的软件产品之前，请做严格的反复的测试和评估
平时升级时，混淆后做常规的测试即可。

混淆后速度影响

混淆后插入的内部算法分支通常增加执行时间5~10%左右
自定义逻辑陷阱等增加约5%弱。
等于说一个100行的代码段，混淆后增加到120行左右，速度影响很微弱。

如何减小文件增加的体积？

在选项中，调大代码切割系数，调高最低代码行数，调小代码扩展空间，均有助于减小文件体积。比如很多uo的祖先，都有一些1-3行代码的代码段，这些毫无混淆的必要，会毫无必要的增大文件空间。在具体使用中，请适当调整和优化配置。

测试版与正式版有何不同？

1. 限制登录时段(视服务器端的设置,也可能不限制)
2. 代码中将被插入一个MessagBox提示窗
3. 处理过的代码只能运行1个月左右。一个月后程序运行时会被终止。

需要做无提示窗口的测试需要来邮件或者msn，qq联系开通临时账号。当然是有购买意向的，无意向的不要随便开来测试。

混淆处理失败或者想再次处理时，无法在PB中进行编译？

下拉选择文件时，对象列表为空白(已经配置好,而且有勾选对象)？

PB编译成功需要文件未被打开，写入，锁定的前提下
请拷贝pbd，dll和exe到一个新的文件夹中进行混淆。一般情况下不要直接对PBL所在的工程文件夹直接操作，另外重新编译pbl文件时请采用full编译方式，因为虽然混淆处理改变了PBD，但源码并未做任何修改，则采用增量编译可能不会重新编译该文件。
PBD被混淆时也一样，如果PBD正在被编译，或者程序正在运行，都无法开启。特别注意。

内部混淆的算法是否固定模式？

在不同时间，不同机器，内部模式会有一些不同，以保持离散性。
另外算法大量采用随机数来选择算法因子。
代码切分后的放置是随机位置插入的。每次不同。
文字的替换，包括变量名，函数名，参数名，对象名等都是采取随机取值。无固定规律。

致谢语

返回目录

对所有购买和测试过的用户,好友,热心人士表示感谢! 感谢你们提出的建议和意见使得软件日趋成熟,能服务更多的人.

谢谢大家的支持!