Pbidea 库关于 PB 核心代码保护的一个方案

我们写程序，总是希望自己的代码不容易被别人破解，有一定安全性。尤其是用 PB 写 CS 程序，如果代码被 pbkiller 之类的工具轻易反编译了，数据连接等信息、加密方法等就一止了然，别人可以直接使用这些信息连接到数据库，其影响不可估量。当然，也有人不在乎这些安全考量，那就不在此考虑之类，可以忽略本文。

一、pbidea 库核心代码安全保护方案

1．给应用程序正常添加库文件。将核心代码单独放到一个 pbl 里面，库名称随意指定，这里假设我们指定的名称是 panda.pbl。这里的“panda”就是关键字。panda.pbl 里必须有个对象叫做 uo_panda，uo_panda 有一个函数 of_init()。

2．正常写程序，把一些初始化代码，包括数据库连接、注意信息等放到 uo_panda 对象或相关对象里面，只要 uo_panda.of_init() 能访问到即可。注意不要在其他 pbl 或 pbd 里包含 panda.pbl 里的对象，否则后面会把 panda.pbl 从库列表里移除，会导致其他对象编译失败。

3．全编译项目，生成 panda.pbd。然后重新设置库列表，把 panda.pbl 从库列表里移除。

类似 panda.pbl 这样的库可以有多个。

4．写这样一段代码，生成加密文件

```
string ls_files[]
ls_files[1] = "panda\panda.pbd"
ls_files[2] = "panda\json" //json 是一个 utf8 编码的 JSON 文件，在调用时会被自动解析到 uo_panda.json 对象里去。当然，你也可以没有这个，可选的。
ls_files[3] = "panda\dzz.png"
```

……..可以将更多需要打包到加密文件里的文件添加进来

uo_utils u

u = create uo_utils

u.packapplication("panda.dat", "mykey", ls_files[]) //生成加密文件 panda.dat

至此，我们得到了一个加密文件包，包含了我们的核心代码。

那么，我们怎么调用这个核心代码呢？其实很简单：

在主窗口的 open 事件里，

uo_utils u

u = create uo_utils

u.initapplication("panda") //注意"panda"这个关键字

initapplication 这个函数加载了加密文件包，并且调用了 uo_panda 对象里的 of_init()函数 ，完成核心代码调用。

二、pbidea 库核心代码安全保护方案的实现原理

本方案主要是利用 aes 256 位加密，key 是随机生成，对库文件进行加密打包，几乎是防止了暴力破解的可能。

Initapplication 函数首先是对文件进行解密，把内容读取到内存，在内存里进行解密。当 PB 程序执行时，分块进行读取，解密成代码后执行。所有 PB 的对象创建、代码运行，都在 DLL 内部完成，这也是 pbidea 库使用 system library 方式的优势。

考虑到可以利用目前各种调试工具，例如 OllyDBG，进行调试跟踪 ，暂停关键代码执行，分析研究程序。Pbidea 库做了一些反调试处理。发现自身处于调试环境中时，会拒绝执行应用功能，从而保护代码安全。即使是调试工具从内存中找到解密后的文件内容，那也只是局部内容，无法保存成整体的文件出来反编译。

大自在 QQ：781770313，QQ 群：624409252

2022 年 4 月 3 日