5种方法，加密你的Python代码

转自：https://baijiahao.baidu.com/s?id=1627375537998184265&wfr=spider&for=pc

Python优点很多，比如简单易学，代码量少，能做的事很多等等，和其他语言一样Pyhton也有一些不可掩盖的缺点，版本不兼容，运行效率不高等等。

其中一个缺点，让不少开发者头疼不已，由于Python解释器开源的关系，导致Python代码无法加密，代码的安全性得不到保障。

当然，想要加密Python代码，也并非无解。最常见的加密方式有4种，还有1种独特的加密方式。

1. 使用pyc

Python解释器在执行代码的过程中，会首先生成.pyc文件，然后再解释执行.pyc中的内容，当然，解释器也能直接执行.pyc文件。

.pyc文件是一个二进制的文件，是不具备可读性的。

假如我们发到客户环境时，是.pyc文件，而不是.py那么是不是就可以保护我们的Python代码？

想要做到这一点，并不难Python标准库就提供了一个名叫compileall的库，使用它就可以做到。

执行如下命令，即可将<src>目录下的所有.py文件编译成.pyc文件：

python -m compileall <src> 然后删除 <src> 目录下所有 .py 文件就可以了。

$ find <src> -name '*.py' -type f -print -exec rm {} \;

这一方法，可以加密我们的Python代码，胜过代码裸在外面。尴尬的是，因为Python解释器的兼容较差，有些版本并不能运行.pyc文件。而且已经有现成的反编译工具，可以直接破解。

比如python-uncompyle6「链接」），只要执行以下命令，就可以搞定。

$ uncompyle6 *compiled-python-file-pyc-or-pyo*

2. 代码混淆

代码混淆，也是一种常见的“ 加密 ”方式，严格意义上说，这一方法并不是加密，而是上代码的可读性变差。比如删除注释，添加毫无意义的注释，添加无效代码，对变量、函数、类进行重命名等。

内容不可读，代码就受到了保护。

代码混淆的工具很多，一个比较好用的混淆库是pyobfuscate GitHub - astrand/pyobfuscate: pyobfuscate这个库可以对类、函数进行重命名，并且插入无关的代码，甚至自动加空格等等。

这一方法很简单，也提高了破解的门槛。但由于代码结构未发生变化，字节码也能获取，破解难度也不高。

一般而言，使用这一方式较为简单，实用。

3. 打包

如果有一款工具，可以将Python脚本打包成在某一平台的可执行文件，最终我们发行的，是一份打包完成的二进制文件，那么程序就更难被破解了？

py2exe FrontPage - py2exe.orgFrontPage - py2exe.org就是一款很好的打包工具，可以将Python脚本打包成可在Windows上运行的文件。

这一方式的优点是进一步提高了破解门槛。遗憾的是，你只能在windows上运行它。

4. Cython

Python运行速度慢何解？用Cython就可以带来性能的提升。实际上Cython也可以用来加密Python代码。

Cython的原理是，将.py编译为.c文件，再将.c文件编译为.so或者.pyd这样一来，文件就变得难以破解了。

这样做的好处是Python代码很难被破解，缺点是有时候Cython可能不支持一小部分代码，完善起来就比较麻烦了。

5. 修改解释器

最后一种方法，做得比较绝。

由于Python是解释型语言，因此在发行Python程序的时候，就必须包含一个Python解释器，如果我们修改这个解释器，代码不就被保护起来了吗？

如果我们能对最原始的Python代码进行加密，加密后的代码被发行后。哪怕被别人看到了，但因为不晓得算法是怎样的，就破解不了了。

这是因为Python解释器本身是一个二进制文件，自然也就无法获得关键性的数据，进而保护了源码。 虽然这一方法最为安全，可操作难度较高。你必须掌握基本的加解密算法，还要探究Python执行代码的方式，从而了解到从什么地方进行加解密。最后禁用字节码，以防通过.pyc反编译即可。

以上五种加密方式，有利有弊，有难有易，根据需求选择就可以了。