Pyinstaller comes with pre-compiled bootloader binaries for different OSs. I suggest compile them by yourself on your machine. Make sure everything is consistent on your machine. For Windows 64bit, install Python 64bit. Download PyInstaller 64bit for Windows. Make sure Visual Studio (VS) corresponding to your Python is installed, check below:

<https://wiki.python.org/moin/WindowsCompilers>

Compile the bootloader of Pyinstaller on your machine with VS. It automatically updates the run.exe, runw.exe, run\_d.exe, runw\_d.exe in DownloadedPyinstallerFolder\PyInstaller\bootloader\Windows-64bit. Check below for more info on how to compile the bootloader:

<https://pyinstaller.readthedocs.io/en/stable/bootloader-building.html>

At the end install Pyinstaller. Within Pyinstaller directory run

python setup.py install

Building the Bootloader

PyInstaller comes with pre-compiled bootloaders for some platforms in the bootloader folder of the distribution folder. When there is no pre-compiled bootloader for the current platform (operating-system and word-size), the [pip](http://www.pip-installer.org/) setup will attempt to build one.

If there is no precompiled bootloader for your platform, or if you want to modify the bootloader source, you need to build the bootloader. To do this,

* Install Visual Studio Comminity
* Download and install Python, which is required for running **waf**,
* *git clone* or download the source (see the Download section on the web-site), <https://github.com/pyinstaller/pyinstaller>
* cd into the folder where you cloned or unpacked the source to,
* cd bootloader, and
* make the bootloader with: python ./waf all,
* test the build by ref:*running (parts of) the test-suite <running-the-test-suite>*.

This will produce the bootloader executables for your current platform (of course, for Windows these files will have the .exe extension):

* ../PyInstaller/bootloader/*OS\_ARCH*/run,
* ../PyInstaller/bootloader/*OS\_ARCH*/run\_d,
* ../PyInstaller/bootloader/*OS\_ARCH*/runw (OS X and Windows only), and
* ../PyInstaller/bootloader/*OS\_ARCH*/runw\_d (OS X and Windows only).

The bootloaders architecture defaults to the machine’s one, but can be changed using the [--target-arch](https://pyinstaller.readthedocs.io/en/stable/usage.html#cmdoption-target-architecture) option – given the appropriate compiler and development files are installed. E.g. to build a 32-bit bootloader on a 64-bit machine, run:

python ./waf all --target-arch=32bit

If this reports an error, read the detailed notes that follow, then ask for technical help.

By setting the environment variable PYINSTALLER\_COMPILE\_BOOTLOADER the [pip](http://www.pip-installer.org/) setup will attempt to build the bootloader for your platform, even if it is already present.

Supported platforms are

* GNU/Linux (using gcc)
* Windows (using Visual C++ (VS2015 or later) or MinGW’s gcc)
* Mac OX X (using clang)