StaticX

Jonathon Reinhart

Jan 16, 2024

CONTENTS

Bundle dynamic executables with their library dependencies so they can be run anywhere, just like a static executable.

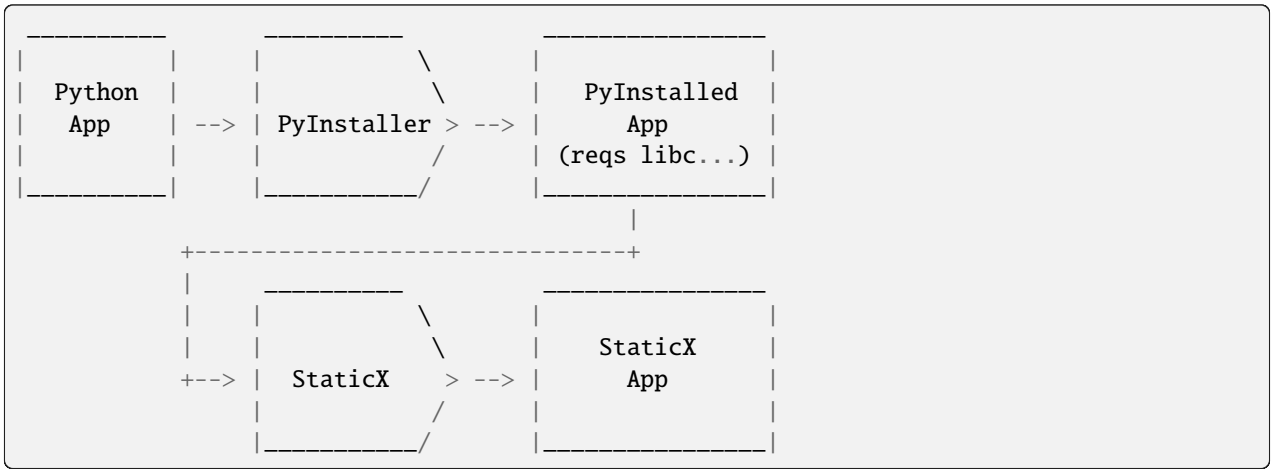
CHAPTER

ONE

INTRODUCTION

StaticX takes a typical dynamic executable and bundles it, along with all of its shared library dependencies, into a single executable. This resulting executable is actually the StaticX *bootloader* with an attached *archive* containing the user executable and libraries. The bootloader will extract the archive to a temporary directory, fix up the executable in its new home, launch it, and clean up after it exits.

StaticX is inspired by PyInstaller, and includes special provisions for working well with PyInstalled executables as input:



CHAPTER

TWO

INSTALLATION

2.1 Requirements

StaticX currently works only with Linux 64-bit dynamic executables.

The following external tools must be installed to use StaticX, all of which are readily available in the package manager of most Linux distributions:

- `ldd` – Part of GNU C Library
- `readelf` – Part of binutils
- `objcopy` – Part of binutils
- `patchelf` – https://github.com/NixOS/patchelf
  - Distro packages available for Debian 8+, Fedora 14+, others (preferred)
  - Or install with `pip install patchelf-wrapper`

The following additional tools must be installed to build StaticX from source:

- `scons`
- musl libc *(optional)*

StaticX is compatible with Python 3.5+.

2.2 Install via pip

StaticX is hosted at PyPI, and the wheels include a bootloader built with musl libc.

Installation via pip is the preferred method:

```
pip3 install staticx
```

2.3 Install from source

If you have musl libc installed, you can use it to build the staticx bootloader, resulting in smaller, better binaries. To do so, set the `BOOTLOADER_CC` environment variable to your `musl-gcc` wrapper path when invoking *pip* or *setup.py*:

```
BOOTLOADER_CC=/usr/local/musl/bin/musl-gcc pip3 install https://github.com/
↪JonathonReinhart/staticx/archive/master.zip
```

Or:

```
cd staticx
BOOTLOADER_CC=/usr/local/musl/bin/musl-gcc pip3 install .
```

CHAPTER

THREE

USAGE

3.1 Synopsis

```
staticx [-h]
        [-l LIB] [--strip] [--no-compress] [-V]
        [--loglevel LEVEL]
        PROG OUTPUT
```

Positional Arguments:

PROG
Input program to bundle

OUTPUT
Output path

Options:

-h, --help	Show help message and exit
-l LIB	Add additional library (absolute path)
	This option can be given multiple times.
--strip	Strip binaries before adding to archive (reduces size)
--no-compress	Don't compress the archive (increases size)
--loglevel LEVEL	Set the logging level (default: WARNING)
	Options: DEBUG,INFO,WARNING,ERROR,CRITICAL
--debug	Set loglevel to DEBUG and use a debug version of the bootloader
-V, --version	Show StaticX version and exit

3.2 Usage

Basic wrapping of an executable:

```
staticx /path/to/exe /path/to/output
```

StaticX will automatically discover and bundle most normal linked libraries. However, libraries loaded by an application at runtime via `dlopen()` cannot currently be detected. These can be manually included in the application bundle by using the `-l` option (any number can be specified by repeating the -l option):

```
staticx -l /path/to/fancy/library.so /path/to/exe /path/to/output
```

3.3 Caveats

StaticX employs a number of tricks to run applications with only their bundled libraries to ensure compatibilitiy. Because of this, there are some caveats that apply to StaticX-bundled applications:

- The dynamic linker is instructed (via `nodeflib`) to only permit bundled libraries to be loaded.
- Target NSS configuration (`/etc/nsswitch.conf`) is ignored (for GLIBC-linked applications) which means that some advanced name services (e.g. Active Directory) will not be available at runtime. For example, looking up the UID number of a domain user will not work.

3.4 Run-time Information

StaticX sets the following environment variables for the wrapped user program:

- `STATICX_BUNDLE_DIR`: The absolute path of the "bundle" directory, the temporary dir where the archive has been extracted.
- `STATICX_PROG_PATH`: The absolute path of the program being executed.

CHAPTER

FOUR

TROUBLESHOOTING

StaticX has to do some weird things in order to work. Thus, you might run into trouble.

4.1 Run-time problems

If, after bundling an application using StaticX, the resulting executable is crashing due to symbol issues or segfaults:

4.1.1 Debug build

The first step is to use the `--debug` flag when bundling:

```
$ staticx --debug myprog myprog.sx
```

This option will:

- Set loglevel to `DEBUG` while building the program
- Use a debug variant of the bootloader which:
  - Adds debug output (to stderr)
  - Includes DWARF debug info

Note: Please include all debugging information if you open an issue.

4.1.2 GDB

If your program segfaults, you can run it under GDB.

Before giving the `r` command to run your program, use `set follow-fork-mode child` so GDB follows the child process.

To get a backtrace:

```
$ gdb --args ./myprog.sx
set follow-fork-mode child
r
(segfault)
bt -full
```

CHAPTER

FIVE

UNSUPPORTED RPATH/RUNPATH

You might be here because you encountered an error like this:

```
staticx: Unsupported PyInstaller input

One or more libraries included in the PyInstaller archive uses unsupported RPATH/RUNPATH␣
↪tags:

  /tmp/staticx-pyi-5ys8gizg/libbar.so: DT_RUNPATH='/bogus/absolute/path'
  /tmp/staticx-pyi-5ys8gizg/libfoo.so: DT_RUNPATH='/bogus/absolute/path'
```

This page will attempt to explain the problem.

5.1 What are RPATH / RUNPATH?

RPATH and RUNPATH are a feature of the GNU dynamic linker/loader. These are entries in the `.dynamic` section which allow a dynamic executable or library to augment or override the preconfigured library path list of the dynamic loader (`ld.so`).

See also: RPATH on Wikipedia

5.2 Why are RPATH / RUNPATH problematic?

These options can circumvent StaticX's ability to maintain positive control of the library search path used by `ld.so`. By doing so, they can allow target-system libraries to be undesirably loaded, possibly causing symbol errors or runtime crashes. After all, StaticX's *modus operandi* is to only execute code included in the bundled archive.

Since StaticX works by setting RPATH on the bundled user executable, there is no need for any of the bundled libraries to use RPATH/RUNPATH. Because of this, StaticX will *strip* RPATH/RUNPATH entries from any library added to the archive. However, StaticX **cannot** modify libraries which are already included in the archive of a PyInstaller application.

5.2.1 RPATH

RPATH is allowed in PyInstaller-included libraries as long as the path is relative to $ORIGIN (the directory where the dynamic executable lives).

If StaticX is complaining about RPATH, it is probably because the library is referencing an absolute path (e.g. /usr/local/lib).

5.2.2 RUNPATH

RUNPATH is even more problematic, because it causes ld.so to **completely disregard** the RPATH set by the StaticX bootloader at program launch. For this reason, RUNPATH is always forbidden.

Unfortunately, this problem is becoming more common since GNU ld now defaults to emitting RUNPATH when the -rpath option is given.

5.3 So what do I do now?

Unfortunately, fixing a problematic PyInstaller-generated executable is not straightforward. The goal is to remove the problematic libraries or replace them with builds which do not use RUNPATH. (A build which uses RPATH is probably acceptable, see above.)

One can sometimes fix this problem by adding --disable-new-dtags to CFLAGS when building/installing a Python package which uses native extensions. The best way to do this is to install them from source into a virtual environment.

5.4 References

- The underlying issue was originally described in #169.
- The auditing check which emits this error was added in #173, and updated in #208.

CHAPTER

SIX

CHANGE LOG

All notable changes to this project will be documented in this file. This project adheres to Semantic Versioning.

6.1 Unreleased

6.1.1 Added

- Added support for Python 3.12 (#272)

6.1.2 Changed

- Introduced `pyproject.toml` and moved metadata from `setup.py` (#267)
- Removed use of deprecated `pkg_resources` (#271, #274)

6.1.3 Fixed

- Fixed an issue with non-ELF "binary" files in PyInstaller archives causing a crash (#270)

6.2 0.14.1 - 2023-08-07

6.2.1 Fixed

- Reverted invalid fix (#255) for libnssfix link failure (#262)
- Fixed issue causing libnssfix link failure when building on GLIBC 2.34 again, by linking against versioned SONAME (.2) (#259)

6.3 0.14.0 - 2023-07-10

This release was pulled from PyPI due to an incorrect fix in #255.

6.3.1 Fixed

- Fixed issue causing libnssfix link failure when building on GLIBC 2.34 (#255)

6.3.2 Removed

- Dropped support for Python 3.5 and 3.6 (#247)

6.4 0.13.9 - 2023-04-16

6.4.1 Fixed

- Updated to support PyInstaller 5.10 archive API changes (#236, #237)
- Refuse to process files with PyInstaller 4.1 or 4.2 (#238)
  - Note: This applies even if the files were built with a different version of PyInstaller; if that version is 4.1 or 4.2, the original bug will manifest.

6.5 0.13.8 - 2022-08-07

6.5.1 Fixed

- Fixed a problem with 0.13.7 release whl (PyPI won't allow re-uploads)

6.6 0.13.7 - 2022-08-07

6.6.1 Fixed

- Fixed an issue where library symlinks with the same basename would present problems (#225)
- Don't crash if .git/ dir is present but git is not installed (#226)
- Fixed potential issue where bootloader linked against glibc could result in target NSS libraries being loaded and causing a cash at startup ([#228])

6.7 0.13.6 - 2021-12-02

6.7.1 Changed

- Change `--debug` option to appear in CLI help output

6.7.2 Fixed

- Fix bug sometimes causing a crash when `-l` is used (#217)

6.8 0.13.5 - 2021-10-26

6.8.1 Fixed

- Handle variables in `DT_NEEDED` tags as seen in `ldd` output on RaspPI OS (#210)

6.9 0.13.4 - 2021-10-22

6.9.1 Changed

- Perform RUNPATH auditing on all PyInstaller archive libraries before aborting (#208)

6.10 0.13.3 - 2021-10-14

6.10.1 Fixed

- Fix ldd warning about libnssfix.so not being executable (#204)

6.11 0.13.2 - 2021-10-09

6.11.1 Added

- Log additional diagnostic information at startup (#199)

6.12 0.13.1 - 2021-10-06

6.12.1 Added

- Log staticx version and arguments at startup (#197)

6.13 0.13.0 - 2021-10-04

6.13.1 Added

- Added auditing of all shared libraries to detect problematic usages of `RPATH/RUNPATH`. Libraries now have `RPATH/RUNPATH` removed while being added, unless those libraries come from a PyInstalled application. (#173)

6.13.2 Changed

- Rework library-adding code to lazily copy libraries before modifying (#192)

6.14 0.12.3 - 2021-09-04

6.14.1 Added

- Added `STATICX_LDD` environment variable to override the `ldd` executable used by Staticx to discover library dependencies. (#180)

6.14.2 Changed

- `LD_LIBRARY_PATH` enviroment variable is now maintained when invoking `ldd` to discover dependencies (#185)

6.15 0.12.2 - 2021-05-22

6.15.1 Fixed

- Worked around patchelf bug which caused `Couldn't find DT_RPATH tag` error at runtime (#175)
- Fixed a bug which caused the glibc hook to crash on non-glibc executables (#179)

6.16 0.12.1 - 2021-02-06

6.16.1 Fixed

- Fixed bug causing libnssfix to be built incorrectly under SCons v4.1.0 (#168)

6.17 0.12.0 - 2020-09-29

6.17.1 Added

- Added support for native 32-bit builds of bootloader (#149)

6.17.2 Changed

- Binary wheels now identify as `manylinux1_x86_64` (#151)

6.17.3 Fixed

- Source distributions build correctly again (#153, #157)

6.17.4 Removed

- Removed more Python 2.7 compatibility cruft (#142, #146, #148)
- Removed additional unnecessary elements of libtar (#154)

6.18 0.11.0 - 2020-07-27

6.18.1 Changed

- Improved tar extraction to minimize number of write() calls (#131)
- Set NODEFLIB flag to prevent any libraries from the target system from being loaded (#138)
- "nssfix" is used to prevent target system `/etc/nssswitch.conf` from being used which would attempt to load system `libnss_*.so` libraries ([#139])

6.18.2 Fixed

- Bundled applications retain their original name (#135)

6.19 0.10.0 - 2020-05-30

6.19.1 Added

- Added `sx-extract` archive extraction/dumping tool (#114)

6.19.2 Removed

- Drop support for Python 2.7 (#115)

6.20 0.9.1 - 2020-01-29

6.20.1 Fixed

- Correct handling of absolute path symlink in ldd output (#118)
- Fixed null tmpdir argument error on GCC9 (#120)
- Fixed ldd "you do not have execution permission…" warning (#122)

6.21 0.9.0 - 2020-01-11

6.21.1 Added

- Staticx binaries now respect `$TMPDIR` for creating temporary directory (#101)

6.21.2 Changed

- Ensure user program is always marked executable in archive (#112)

6.21.3 Fixed

- Don't hard-code exclusion of `linux-vdso.so.1` (#102)

6.21.4 Removed

- Drop support for Python 3.4 (#111)

6.22 0.8.1 - 2019-12-30

6.22.1 Changed

- Changed `setup.py` to respect `BOOTLOADER_CC`, to simplify `.travis.yml` and ensure that released wheels are always built with musl-libc.

6.23 0.8.0 - 2019-12-30

6.23.1 Added

- Set `STATICX_BUNDLE_DIR` and `STATICX_PROG_PATH` in child process (#81)
- Add `--debug` flag to bundle debug bootloader (#87)

6.23.2 Changed

- Changed pyinstaller hook to ignore static executables (#83)

6.23.3 Fixed

- Always generate tar archive in GNU format. Python 3.8 changed the default to PAX which is not supported by our libtar. (#85)
- Add backports.lzma to setup.py for Python 2, removing manual requirement (#89)

6.24 0.7.0 - 2019-03-10

6.24.1 Changed

- Refactored and trimmed libtar (#74)

6.24.2 Fixed

- Correctly handle applications and libraries that specify an `RPATH` including `$ORIGIN`, including apps built with PyInstaller (#75)
- Fixed potential issue in ignored libraries list (#77)
- Fixed missing libxz in source distributions (#77)

6.25 0.6.0 - 2018-11-13

6.25.1 Added

- Add `--no-compress` option to store archive uncompressed (#58)

6.25.2 Changed

- Detect if user app is a different machine type than the bootloader (#56)
- Drop support for Python 3.2 and 3.3 (#65)

6.25.3 Fixed

- Use <sys/sysmacros.h> for makedev() (#63)
- Handle subdirectories when extracting Pyinstaller archives (#69)
- Handle shared objects with no dependencies (#70)

6.26 0.5.0 - 2017-07-16

6.26.1 Added

- Added `--strip` option to strip binaries while adding to archive (#39)

6.26.2 Changed

- Raise error if given output path is a directory (#52)
- Dynamically select XZ BCJ filter (#54)

6.27 0.4.1 - 2017-07-15

6.27.1 Fixed

- Fixes for release builds deployed to PyPI

6.28 0.4.0 - 2017-07-13

6.28.1 Added

- Compress archive with LZMA (plus x86 BCJ filter) (#46)

6.29 0.3.2 - 2017-06-15

6.29.1 Fixed

- Fixed PyPI bdists not including bootloader (#32)

6.30 0.3.1 - 2017-06-14

6.30.1 Fixed

- Work around `FTW_MOUNT` bug in musl<1.0.0 (#30)

6.31 0.3.0 - 2017-06-13

6.31.1 Added

- Auto-detect additional dependencies for apps built with PyInstaller (#21)

6.31.2 Changed

- Compatibility fixes for older versions of Python and GCC
- Handle multiple levels of library symlinks (#18)

6.32 0.2.0 - 2017-05-31

6.32.1 Changed

- Work on temporary file while building application (#12)
- Run user application in child process to enable cleanup (#9)

6.33 0.1.0 - 2017-05-30

Initial release