pylibacl Documentation

Release 0.7.0

Iustin Pop

Apr 24, 2023

Contents

See the *README* for start, or the detailed *module* information.

CHAPTER 1

Contents

1.1 pylibacl

This is a Python 3.7+ extension module allows you to manipulate the POSIX.1e Access Control Lists present in some OS/file-systems combinations.

Downloads: go to https://pylibacl.k1024.org/downloads. Latest version is 0.7.0. The source repository is either at https://git.k1024.org/pylibacl.git or at https://github.com/iustin/pylibacl.

For any issues, please file bugs at https://github.com/iustin/pylibacl/issues.

See the `CONTRIBUTING.md` file for details on how to contribute.

GitHub Workflow Status Codecov Read the Docs GitHub issues GitHub tag (latest by date) GitHub release (latest by date) PyPI Debian package Ubuntu package GitHub Release Date GitHub commits since latest release GitHub last commit

1.1.1 Requirements

pylibacl has been written and tested on Linux, kernel v2.4 or newer, with XFS filesystems; ext2/ext3 should also work. Since release 0.4.0, FreeBSD 7 also has quite good support. If any other platform implements the POSIX.1e draft, pylibacl can be used. I heard that Solaris does, but I can't test it.

- Python 3.7 or newer. Python 2.4+ was supported in the 0.5.x branch, Python 3.4+ in the 0.6 branch.
- Operating system:
  - Linux, kernel v2.4 or newer, and the libacl library and development packages (all modern distributions should have this, under various names); also the file-systems you use must have ACLs turned on, either as a compile or mount option.
  - FreeBSD 7.0 or newer.
- The sphinx python module, for your python version, if building the documentation.

1.1.2 FreeBSD

Note that on FreeBSD, ACLs are not enabled by default (at least on UFS file systems). To enable them, run `tunefs -a enabled` on the file system in question (after mounting it read-only). Then install:

- `pkg install py36-setuptools py36-sphinx`

or:

- `pkg install py37-setuptools`

1.1.3 Security

For reporting security vulnerabilities, please see `SECURITY.md`.

1.1.4 License

pylibacl is Copyright (C) 2002-2009, 2012, 2014, 2015 Iustin Pop.

pylibacl is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. See the COPYING file for the full license terms.

1.2 Contributing to pylibacl

Hi, and thanks for any and all contributions!

1.2.1 Bugs and patches

This is a small project, so let's keep things simple:

- Please file all bug reports on github (https://github.com/iustin/pylibacl/issues), as this allows archival and discovery by other people;
- Send patches as pull requests; for larger changes, would be good to first open a bug to discuss the plans;

Due to simplicity, there are no old branches being kept alive, but if it ever happens that a bug is found in older versions and there is needed to support older Python versions, it is possible to do so.

1.2.2 Code standards

There are no formal standards, but:

- Code should be tested - this is why there's a Codecov integration.
- New functions should have good docstrings (in the C code).
- New functions/constants should be listed in the documentation, see `doc/module.rst` for how to include them.
- All non-trivial changes should be listed in `NEWS.md` for further inclusion in new releases documentation. Add an "unreleased" section (if one doesn't exist yet) to list the changes.

1.2.3 Release process

Right now, due to GPG signing, I'm doing releases and signing them manually (offline, I mean). Basically, once GitHub workflows are fine:

- Bump the version in all places - use `git grep -F $OLD_VER` and update as needed.
- Ensure that `setup.py` has the right Python versions listed (bit me more than once).
- Update the `NEWS.md` file is up to date (contents), and use the right date.
- Check that the generated documentation (`make doc`) looks right.

Then run these steps:

```
$ make clean
$ make distcheck # this leaves things in dist/
$ git tag -m 'Release pylibacl-0.0.1' --sign v0.0.1
$ gpg --sign -b -a dist/pylibacl-0.0.1.tar.gz
$ python3 -m twine upload dist/*
```

Separately:

- Upload the `dist/` contents to GitHub and tag a new release.
- Upload the `dist/` contents to the old-style download area, https://pylibacl.k1024.org/downloads/.

Hopefully one day all this can be more automated.

1.2.4 Signing key

The releases are currently signed by my key, see https://k1024.org/contact/.

1.3 Security Policy

To report a (potential or confirmed) security issue, please email iustin@k1024.org with a description of the issue, steps to reproduce it, affected versions, and if known, mitigations for the issue.

Since this is a small project, there's no list of supported versions. I will attempt to reply to reports within a working week, and to fix and disclose vulnerabilities within 90 days, but this is not a guarantee.

Optionally, you can encrypt the email with my GPG key, see for details https://k1024.org/contact/.

Alternatively, you can use the GitHub "Private vulnerability reporting" functionality (but note this is beta).

1.4 Implementation details

1.4.1 Functionality level

The IEEE 1003.1e draft 17 ("POSIX.1e") describes a set of 28 functions. These are grouped into three groups, based on their portability:

- first group, the most portable one. All systems which claim to support POSIX.1e should implement these:

  acl_delete_def_file(3), acl_dup(3), acl_free(3), acl_from_text(3), acl_get_fd(3), acl_get_file(3), acl_init(3), acl_set_fd(3), acl_set_file(3), acl_to_text(3), acl_valid(3)

- second group, containing the rest of the POSIX ACL functions. Systems which claim to fully implement POSIX.1e should implement these:

  acl_add_perm(3), acl_calc_mask(3), acl_clear_perms(3), acl_copy_entry(3), acl_copy_ext(3), acl_copy_int(3), acl_create_entry(3), acl_delete_entry(3), acl_delete_perm(3), acl_get_entry(3), acl_get_permset(3), acl_get_qualifier(3), acl_get_tag_type(3), acl_set_permset(3), acl_set_qualifier(3), acl_set_tag_type(3), acl_size(3)

- third group, containing extra functions implemented by each OS. These are non-portable version. Both Linux and FreeBSD implement some extra functions.

Thus we have the level of compliance. Depending on whether the system library support the second group, you get some extra methods for the ACL object.

The implementation of the second group of function can be tested by checking the module-level constant HAS_ACL_ENTRY. The extra functionality available on Linux can be tested by additional HAS_* constants.

1.4.2 Internal structure

The POSIX draft has the following stuff (correct me if I'm wrong):

- an ACL is denoted by acl_t
- an ACL contains many acl_entry_t, these are the individual entries in the list; they always(!) belong to an acl_t
- each entry_t has a qualifier (think uid_t or gid_t), whose type is denoted by the acl_tag_t type, and an acl_permset_t
- the acl_permset_t can contain acl_perm_t value (ACL_READ, ACL_WRITE, ACL_EXECUTE, ACL_ADD, ACL_DELETE, . . .)
- functions to manipulate all these, and functions to manipulate files

1.4.3 Currently supported platforms

For any other platforms, volunteers are welcome.

Linux

It needs kernel 2.4 or higher and the libacl library installed (with development headers, if installing from rpm). This library is available on all modern distributions.

The level of compliance is level 2 (see IMPLEMENTATION), plus some extra functions; and as my development is done on Linux, I try to implement these extensions when it makes sense.

FreeBSD

The current tested version is 7.0. FreeBSD supports all the standards functions, but 7.0-RELEASE seems to have some issues regarding the acl_valid() function when the qualifier of an ACL_USER or ACL_GROUP entry is the same as the current uid. By my interpretation, this should be a valid ACL, but FreeBSD declares the ACL invalid. As such, some unittests fail on FreeBSD.

1.4.4 Porting to other platforms

First, determine if your OS supports the full 28 functions of the POSIX.1e draft (if so, define HAVE_LEVEL2) or only the first 11 functions (most common case, meaning only HAVE_LEVEL1).

If your OS supports only LEVEL1, modify `setup.py` as appropriately; unfortunately, the functionality of the module is quite low.

If your OS supports LEVEL2, there is a function which you must define: testing if an acl_permset_t contains a given permission. For example, under Linux, the acl library defines:

```
int acl_get_perm(acl_permset_t permset_d, acl_perm_t perm);
```

under FreeBSD, the library defines `acl_get_perm_np` with a similar syntax. So just see how this is implemented in your platform and either define a simple macro or a full function with the syntax:

```
static int get_perm(acl_permset_t permset_d, acl_perm_t perm);
```

which must return 1 if the permset contains perm and 0 otherwise.

1.5 News

1.5.1 Version 0.7.0

released Sun, 23 Apr 2023

Important: Python 3.7 is the minimum supported version, due to difficulty of testing old releases, and the fact that everything older has been deprecated a long time ago (e.g. 3.6 at the end of 2021).

Otherwise, a minor release:

- Improve error handling in some corner cases (not expected to have any real-life impact, but who knows).
- Improved testing coverage and test infrastructure.
- Modernise parts of the C code based on recent Python version guidelines.
- Add a simple security policy and contribution guidelines.

1.5.2 Version 0.6.0

released Sun, 29 Nov 2020

Major release removing Python 2 support. This allow both code cleanup and new features, such as:

- Support for pathlib objects in `apply_to` and `has_extended` functions when running with Python 3.6 and newer.
- Use of built-in C API functions for bytes/unicode/pathlib conversion when dealing with file names, removing custom code (with the associated benefits).

Important API changes/bug fixes:

- Initialisation protocol has been changed, to disallow uninitialised objects; this means that `__new__` will always create valid objects, to prevent the need for checking initialisation status in all code paths; this also (implicitly) fixes memory leaks on re-initialisation (calling `__init__(...)` on an existing object) and segfaults (!) on non-initialised object attribute access. Note ACL re-initialisation is tricky and (still) leads to undefined behaviour of existing Entry objects pointing to it.

- Fix another bug in ACL re-initialisation where failures would result in invalid objects; now failed re-initialisation does not touch the original object.
- Restore `__setstate__`/`__getstate__` support on Linux; this was inadvertently removed due a typo(!) when adding support for it in FreeBSD. Pickle should work again for ACL instances, although not sure how stable this serialisation format actually is.
- Additionally, slightly change `__setstate__()` input to not allow Unicode, since the serialisation format is an opaque binary format.
- Fix (and change) entry qualifier (which is a user/group ID) behaviour: assume/require that uid_t/gid_t are unsigned types (they are with glibc, MacOS and FreeBSD at least; the standard doesn't document the signedness), and convert parsing and returning the qualifier to behave accordingly. The breakage was most apparent on 32-bit architectures, in which context the problem was originally reported (see issue #13).

Minor improvements:

- Added a `data` keyword argument to `ACL()`, which allows restoring an ACL directly from a serialised form (as given by `__getstate__()`), which should simplify some uses cases (`a = ACL(); a.__setstate__(...)`).
- When available, add the file path to I/O error messages, which should lead to easier debugging.
- The test suite has changed to `pytest`, which allows increased coverage via parameterisation.

1.5.3 Version 0.5.4

released Thu, 14 Nov 2019

Maintenance release:

- Switch build system to Python 3 by default (can be overridden if needed).
- Internal improvements for better cpychecker support.
- Fix compatibility with PyPy.
- Test improvements (both local and on Travis), testing more variations (debug, PyPy).
- Improve test coverage, and allow gathering test coverage results.
- Drop support (well, drop testing) for Python lower than 2.7.
- Minor documentation improvements (closes #9, #12).

1.5.4 Version 0.5.3

released Thu, 30 Apr 2015

FreeBSD fixes:

- Enable all FreeBSD versions after 7.x at level 2 (thanks to Garrett Cooper).
- Make test suite pass under FreeBSD, which has a stricter behaviour with regards to invalid ACLs (which we do exercise in the test suite), thanks again to Garret for the bug reports.

1.5.5 Version 0.5.2

released Sat, 24 May 2014

No visible changes release: just fix tests when running under pypy.

1.5.6 Version 0.5.1

released Sun, 13 May 2012

A bug-fix only release. Critical bugs (memory leaks and possible segmentation faults) have been fixed thanks to Dave Malcolm and his `cpychecker` tool. Additionally, some compatibility issues with Python 3.x have been fixed (str() methods returning bytes).

The documentation has been improved and changed from epydoc to sphinx; note however that the documentation is still auto-generated from the docstrings.

Project reorganisation: the project home page has been moved from SourceForge to GitHub.

1.5.7 Version 0.5

released Sun, 27 Dec 2009

Added support for Python 3.x and improved support for Unicode filenames.

1.5.8 Version 0.4

released Sat, 28 Jun 2008

License

Starting with this version, pylibacl is licensed under LGPL 2.1, Febryary 1999 or any later versions (see README.rst and COPYING).

Linux support

A few more Linux-specific functions:

- add the ACL.equiv_mode() method, which will return the equivalent octal mode if this is a basic ACL and raise an IOError exception otherwise
- add the acl_extended(…) function, which will check if an fd or path has an extended ACL

FreeBSD support

FreeBSD 7.x will have almost all the acl manipulation functions that Linux has, with the exception of **getstate/setstate**. As a workaround, use the str() and ACL(text=…) methods to pass around textual representations.

Interface

At module level there are now a few constants exported for easy-checking at runtime what features have been compiled in:

- `HAS_ACL_FROM_MODE`, denoting whether the ACL constructor supports the `mode=0xxx` parameter
- `HAS_ACL_CHECK`, denoting whether ACL instances support the `check()` method
- `HAS_ACL_ENTRY`, denoting whether ACL manipulation is possible and the Entry and Permset classes are available

- `HAS_EXTENEDED_CHECK`, denoting whether the `acl_extended()` function is supported
- `HAS_EQUIV_MODE`, denoting whether ACL instances support the `equiv_mode()` method

Internals

Many functions have now unittests, which is a good thing.

1.5.9 Version 0.3

released Sun, 21 Oct 2007

Linux support

Under Linux, implement more functions from libacl:

- add `ACL(mode=...)`, implementing `acl_from_mode`.
- add `ACL.to_any_text()`, implementing `acl_to_any_text`.
- add ACL comparison, using `acl_cmp`.
- add `ACL.check()`, which is a more descriptive function than validate.

Also see the search.