manhole

Release 1.5.0

Dec 29, 2017

Contents

CHAPTER 1

Overview

docs	
tests	
package	

Manhole is in-process service that will accept unix domain socket connections and present the stacktraces for all threads and an interactive prompt. It can either work as a python daemon thread waiting for connections at all times *or* a signal handler (stopping your application and waiting for a connection).

Access to the socket is restricted to the application's effective user id or root.

This is just like Twisted's manhole. It's simpler (no dependencies), it only runs on Unix domain sockets (in contrast to Twisted's manhole which can run on telnet or ssh) and it integrates well with various types of applications.

Documentation http://python-manhole.readthedocs.org/en/latest/

1.1 Usage

Install it:

```
pip install manhole
```

You can put this in your django settings, wsgi app file, some module that's always imported early etc:

```
import manhole
manhole.install() # this will start the daemon thread

# and now you start your app, eg: server.serve_forever()
```

Now in a shell you can do either of these:

```
netcat -U /tmp/manhole-1234
socat - unix-connect:/tmp/manhole-1234
socat readline unix-connect:/tmp/manhole-1234
```

Socat with readline is best (history, editing etc).

Sample output:

```
$ nc -U /tmp/manhole-1234

Python 2.7.3 (default, Apr 10 2013, 06:20:15)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> dir()
['__builtins__', 'dump_stacktraces', 'os', 'socket', 'sys', 'traceback']
>>> print 'foobar'
foobar
```

1.1.1 Alternative client

There's a new experimental `manhole-cli` bin since 1.1.0, that emulates `socat`:

```
usage: manhole-cli [-h] [-t TIMEOUT] [-1 | -2 | -s SIGNAL] PID

Connect to a manhole.

positional arguments:
  PID                   A numerical process id, or a path in the form:
                        /tmp/manhole-1234

optional arguments:
  -h, --help            show this help message and exit
  -t TIMEOUT, --timeout TIMEOUT
                        Timeout to use. Default: 1 seconds.
  -1, -USR1             Send USR1 (10) to the process before connecting.
  -2, -USR2             Send USR2 (12) to the process before connecting.
  -s SIGNAL, --signal SIGNAL
                        Send the given SIGNAL to the process before
                        connecting.
```

1.2 Features

- Uses unix domain sockets, only root or same effective user can connect.
- Can run the connection in a thread or in a signal handler (see `oneshot_on` option).
- Can start the thread listening for connections from a signal handler (see `activate_on` option)

- Compatible with apps that fork, reinstalls the Manhole thread after fork - had to monkeypatch os.fork/os.forkpty for this.
- Compatible with gevent and eventlet with some limitations - you need to either:
  - Use `oneshot_on`, *or*
  - Disable thread monkeypatching (eg: `gevent.monkey.patch_all(thread=False)`, `eventlet.monkey_patch(thread=False)`

  Note: on eventlet you might need to setup the hub first to prevent circular import problems:

```
import eventlet
eventlet.hubs.get_hub()  # do this first
eventlet.monkey_patch(thread=False)
```

- The thread is compatible with apps that use signalfd (will mask all signals for the Manhole threads).

1.2.1 Options

```
manhole.install(
    verbose=True,
    verbose_destination=2,
    patch_fork=True,
    activate_on=None,
    oneshot_on=None,
    sigmask=manhole.ALL_SIGNALS,
    socket_path=None,
    reinstall_delay=0.5,
    locals=None,
    strict=True,
)
```

- `verbose` - Set it to `False` to squelch the logging.
- `verbose_destination` - Destination for verbose messages. Set it to a file descriptor or handle. Default is unbuffered stderr (stderr `2` file descriptor).
- `patch_fork` - Set it to `False` if you don't want your `os.fork` and `os.forkpy` monkeypatched
- `activate_on` - Set to `"USR1"`, `"USR2"` or some other signal name, or a number if you want the Manhole thread to start when this signal is sent. This is desirable in case you don't want the thread active all the time.
- `thread` - Set to `True` to start the always-on ManholeThread. Default: `True`. Automatically switched to `False` if `oneshot_on` or `activate_on` are used.
- `oneshot_on` - Set to `"USR1"`, `"USR2"` or some other signal name, or a number if you want the Manhole to listen for connection in the signal handler. This is desireable in case you don't want threads at all.
- `sigmask` - Will set the signal mask to the given list (using `signalfd.sigprocmask`). No action is done if `signalfd` is not importable. **NOTE**: This is done so that the Manhole thread doesn't *steal* any signals; Normally that is fine because Python will force all the signal handling to be run in the main thread but signalfd doesn't.
- `socket_path` - Use a specific path for the unix domain socket (instead of `/tmp/manhole-<pid>`). This disables `patch_fork` as children cannot reuse the same path.
- `reinstall_delay` - Delay the unix domain socket creation *reinstall_delay* seconds. This alleviates cleanup failures when using fork+exec patterns.
- `locals` - Names to add to manhole interactive shell locals.

- `daemon_connection` - The connection thread is daemonic (dies on app exit). Default: `False`.
- `redirect_stderr` - Redirect output from stderr to manhole console. Default: `True`.
- `strict` - If `True` then `AlreadyInstalled` will be raised when attempting to install manhole twice. Default: `True`.

1.2.2 Environment variable installation

Manhole can be installed via the `PYTHONMANHOLE` environment varialbe.

This:

```
PYTHONMANHOLE='' python yourapp.py
```

Is equivalent to having this in `yourapp.py`:

```
import manhole
manhole.install()
```

Any extra text in the environment variable is passed to `manhole.install()`. Example:

```
PYTHONMANHOLE='onshot_on="USR2"' python yourapp.py
```

1.2.3 What happens when you actually connect to the socket

1. Credentials are checked (if it's same user or root)
2. `sys.__std*__`/`sys.std*` are redirected to the UDS
3. Stacktraces for each thread are written to the UDS
4. REPL is started so you can fiddle with the process

1.3 Known issues

- Using threads and file handle (not raw file descriptor) `verbose_destination` can cause deadlocks. See bug reports: PyPy and Python 3.4.

1.3.1 SIGTERM and socket cleanup

By default Python doesn't call the `atexit` callbacks with the default SIGTERM handling. This makes manhole leave stray socket files around. If this is undesirable you should install a custom SIGTERM handler so `atexit` is properly invoked.

Example:

```
import signal
import sys

def handle_sigterm(signo, frame):
    sys.exit(128 + signo)  # this will raise SystemExit and cause atexit to be called

signal.signal(signal.SIGTERM, handle_sigterm)
```

1.3.2 Using Manhole with uWSGI

Because uWSGI overrides signal handling Manhole is a bit more tricky to setup. One way is to use "uWSGI signals" (not the POSIX signals) and have the workers check a file for the pid you want to open the Manhole in.

Stick something this in your WSGI application file:

```
from __future__ import print_function
import sys
import os
import manhole

stack_dump_file = '/tmp/manhole-pid'
uwsgi_signal_number = 17

try:
    import uwsgi

    if not os.path.exists(stack_dump_file):
        open(stack_dump_file, 'w')

    def open_manhole(dummy_signum):
        with open(stack_dump_file, 'r') as fh:
            pid = fh.read().strip()
            if pid == str(os.getpid()):
                inst = manhole.install(strict=False, thread=False)
                inst.handle_oneshot(dummy_signum, dummy_signum)

    uwsgi.register_signal(uwsgi_signal_number, 'workers', open_manhole)
    uwsgi.add_file_monitor(uwsgi_signal_number, stack_dump_file)

    print("Listening for stack mahole requests via %r" % (stack_dump_file,), file=sys.
↪stderr)
except ImportError:
    print("Not running under uwsgi; unable to configure manhole trigger", file=sys.
↪stderr)
except IOError:
    print("IOError creating manhole trigger %r" % (stack_dump_file,), file=sys.stderr)


# somewhere bellow you'd have something like
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
# or
def application(environ, start_response):
    start_response('200 OK', [('Content-Type', 'text/plain'), ('Content-Length', '2
↪')])
    yield b'OK'
```

To open the Manhole just run *echo 1234 > /tmp/manhole-pid* and then *manhole-cli 1234*.

1.4 Requirements

OS Linux, OS X

Runtime Python 2.7, 3.4, 3.5, 3.6 or PyPy

1.5 Similar projects

- Twisted's manhole - it has colors and server-side history.
- wsgi-shell - spawns a thread.
- pyrasite - uses gdb to inject code.
- pydbattach - uses gdb to inject code.
- pystuck - very similar, uses rpyc for communication.
- pyringe - uses gdb to inject code, more reliable, but relies on *dbg* python builds unfortunatelly.
- pdb-clone - uses gdb to inject code, with a different strategy.

CHAPTER 2

Installation

At the command line:

```
pip install manhole
```

CHAPTER 3

Usage

To use manhole in a project:

```
import manhole
```

CHAPTER 4

Reference

4.1 manhole

class manhole.**Logger**
Internal object used for logging.

Initially this is not configured. Until you call `manhole.install()`, this logger object won't work (will raise `NotInstalled`).

time() → floating point number
Return the current time in seconds since the Epoch. Fractions of a second may be present if the system clock provides them.

class manhole.**ManholeConnectionThread**(*client*, *connection_handler*, *daemon=False*)
Manhole thread that handles the connection. This thread is a normal thread (non-daemon) - it won't exit if the main thread exits.

class manhole.**ManholeThread**(*get_socket*, *sigmask*, *start_timeout*, *connection_handler*, *bind_delay=None*, *daemon_connection=False*)
Thread that runs the infamous "Manhole". This thread is a *daemon* thread - it will exit if the main thread exits.

On connect, a different, non-daemon thread will be started - so that the process won't exit while there's a connection to the manhole.

Parameters

- **sigmask** (*list of signal numbers*) – Signals to block in this thread.
- **start_timeout** (*float*) – Seconds to wait for the thread to start. Emits a message if the thread is not running when calling `start()`.
- **bind_delay** (*float*) – Seconds to delay socket binding. Default: *no delay*.
- **daemon_connection** (*bool*) – The connection thread is daemonic (dies on app exit). Default: `False`.

clone(***kwargs*)
Make a fresh thread with the same options. This is usually used on dead threads.

run()
Runs the manhole loop. Only accepts one connection at a time because:

- This thread is a daemon thread (exits when main thread exists).
- The connection need exclusive access to stdin, stderr and stdout so it can redirect inputs and outputs.

manhole.**check_credentials**(*client*)
Checks credentials for given socket.

manhole.**dump_stacktraces**()
Dumps thread ids and tracebacks to stdout.

manhole.**get_peercred**(*sock*)
Gets the (pid, uid, gid) for the client on the given *connected* socket.

manhole.**handle_connection_exec**(*client*)
Alternate connection handler. No output redirection.

manhole.**handle_connection_repl**(*client*)
Handles connection.

manhole.**handle_repl**(*locals*)
Dumps stacktraces and runs an interactive prompt (REPL).

manhole.**install**(*verbose=True*, *verbose_destination=2*, *strict=True*, ***kwargs*)
Installs the manhole.

Parameters

- **verbose** (*bool*) – Set it to `False` to squelch the logging.
- **verbose_destination** (*file descriptor or handle*) – Destination for verbose messages. Default is unbuffered stderr (stderr `2` file descriptor).
- **patch_fork** (*bool*) – Set it to `False` if you don't want your `os.fork` and `os.forkpy` monkeypatched
- **activate_on** (*int or signal name*) – set to `"USR1"`, `"USR2"` or some other signal name, or a number if you want the Manhole thread to start when this signal is sent. This is desireable in case you don't want the thread active all the time.
- **oneshot_on** (*int or signal name*) – Set to `"USR1"`, `"USR2"` or some other signal name, or a number if you want the Manhole to listen for connection in the signal handler. This is desireable in case you don't want threads at all.
- **thread** (*bool*) – Start the always-on ManholeThread. Default: `True`. Automatically switched to `False` if `oneshort_on` or `activate_on` are used.
- **sigmask** (*list of ints or signal names*) – Will set the signal mask to the given list (using `signalfd.sigprocmask`). No action is done if `signalfd` is not importable. **NOTE**: This is done so that the Manhole thread doesn't *steal* any signals; Normally that is fine because Python will force all the signal handling to be run in the main thread but signalfd doesn't.
- **socket_path** (*str*) – Use a specific path for the unix domain socket (instead of `/tmp/manhole-<pid>`). This disables `patch_fork` as children cannot reuse the same path.
- **reinstall_delay** (*float*) – Delay the unix domain socket creation *reinstall_delay* seconds. This alleviates cleanup failures when using fork+exec patterns.
- **locals** (*dict*) – Names to add to manhole interactive shell locals.

- **daemon_connection** (*bool*) – The connection thread is daemonic (dies on app exit). Default: `False`.
- **redirect_stderr** (*bool*) – Redirect output from stderr to manhole console. Default: `True`.
- **connection_handler** (*function*) – Connection handler to use. Use `"exec"` for simple implementation without output redirection or your own function. (warning: this is for advanced users). Default: `"repl"`.

CHAPTER 5

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

5.1 Bug reports

When reporting a bug please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.2 Documentation improvements

Manhole could always use more documentation, whether as part of the official manhole docs, in docstrings, or even on the web in blog posts, articles, and such.

5.3 Feature requests and feedback

The best way to send feedback is to file an issue at https://github.com/ionelmc/python-manhole/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that code contributions are welcome :)

5.4 Development

To set up *python-manhole* for local development:

1. Fork python-manhole (look for the "Fork" button).
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/python-manhole.git
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. Commit your changes and push your branch to GitHub:

```
git add .
git commit -m "Your detailed description of your changes."
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

5.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`)[1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

5.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 6

Authors

- Ionel Cristian Mărieș - http://blog.ionelmc.ro
- Saulius Menkevičius - https://github.com/razzmatazz
- Nir Soffer - http://nirs.freeshell.org

CHAPTER 7

Changelog

7.1 1.5.0 (2017-08-31)

- Added two string aliases for `connection_handler` option. Now you can conveniently use `connection_handler="exec"`.
- Improved `handle_connection_exec`. It now has a clean way to exit (`exit()`) and properly closes the socket.

7.2 1.4.0 (2017-08-29)

- Added the `connection_handler` install option. Default value is `manhole.handle_connection_repl`, and alternate `manhole.handle_connection_exec` is provided (very simple: no output redirection, no stacktrace dumping).
- Dropped Python 3.2 from the test grid. It may work but it's a huge pain to support (pip/pytest don't support it anymore).
- Added Python 3.5 and 3.6 in the test grid.
- Fixed issues with piping to `manhole-cli`. Now `echo foobar | manhole-cli` will wait 1 second for output from manhole (you can customize this with the `--timeout` option).
- Fixed issues with newer PyPy (caused by gevent/eventlet socket unwrapping).

7.3 1.3.0 (2015-09-03)

- Allowed Manhole to be configured without any thread or activation (in case you want to manually activate).
- Added an example and tests for using Manhole with uWSGi.
- Fixed error handling in `manhole-cli` on Python 3 (exc vars don't leak anymore).

- Fixed support for running in gevent/eventlet-using apps on Python 3 (now that they support Python 3).
- Allowed reinstalling the manhole (in non-`strict` mode). Previous install is undone.

7.4 1.2.0 (2015-07-06)

- Changed `manhole-cli`:
  - Won't spam the terminal with errors if socket file doesn't exist.
  - Allowed sending any signal (new `--signal` argument).
  - Fixed some validation issues for the `PID` argument.

7.5 1.1.0 (2015-06-06)

- Added support for installing the manhole via the `PYTHONMANHOLE` environment variable.
- Added a `strict` install option. Set it to false to avoid getting the `AlreadyInstalled` exception.
- Added a `manhole-cli` script that emulates `socat readline unix-connect:/tmp/manhole-1234`.

7.6 1.0.0 (2014-10-13)

- Added `socket_path` install option (contributed by Nir Soffer).
- Added `reinstall_delay` install option.
- Added `locals` install option (contributed by Nir Soffer).
- Added `redirect_stderr` install option (contributed by Nir Soffer).
- Lots of internals cleanup (contributed by Nir Soffer).

7.7 0.6.2 (2014-04-28)

- Fix OS X regression.

7.8 0.6.1 (2014-04-28)

- Support for OS X (contributed by Saulius Menkevičius).

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index