
Onion http server library

 Travis status

 Coverity status

Onion is a C library to create simple HTTP servers and Web Applications.

[master](#) the development branch. Current stable branch is [onion-0-8](#).

Introduction

The use case is an existing application, or a new one, that needs some HTTP interconnection with the world. It uses the library to add some handlers for specific URLs and generate and serve the dynamic data as needed.

It also has security goals (SSL support) so that you just concentrate on what you want to serve, and serve it.

Its not a web server per se, as it is not an executable.

If you want to compare to a web server, a web server would use a module or plugin to add some functionality. With libonion you have the functionality and add the webserver as a plugin.

There is a wiki available at <https://github.com/davidmoreno/onion/wiki>, with useful information on how to get started using Onion and it's internal workings.

API documentation is at <http://coralbits.com/static/onion/>.

There is a mailing list at <https://groups.google.com/a/coralbits.com/forum/?fromgroups=#!forum/onion-dev>

Collaborate!

You can, and are encouraged, to branch at github, download and tweak onion to use it in your projects.

The library is dual licensed under the Apache2 license and GPLv2+, so you can make almost anything with it, use it in your commercial and free software programs, and modify it to your needs.

Please join the mailing list at <https://groups.google.com/a/coralbits.com/group/onion-dev/topics>, to ask your questions and comment on your success using onion.

There is also a blog to keep everybody informed about news on onion at <http://blog.coralbits.com/>.

Download

There are third party packages available:

- RPM based: <http://software.opensuse.org/download.html?project=home%3Admoreno&package=onion>
- Raspberry pi: <http://packages.aisoy.com/debian/pool/unstable/libo/libonion/>
- Arch Linux: <https://aur.archlinux.org/packages/libonion-git/>

If you know of any other packaged version, please send me a note.

As always they may be outdated, if you want the latest and greatest, do a manual compile and install.

Thanks to Ruediger Meier for helping for so long with the RPM packages.

Compile and Install

Manual compile and install:

```
1 $ git clone git@github.com:davidmoreno/onion.git
2 $ cd onion
3 $ mkdir build
4 $ cd build
5 $ cmake ..
6 $ make
7 $ sudo make install
```

To compile with debugging enabled, use

```
1 $ cmake -DCMAKE_BUILD_TYPE=Debug ..
```

To run with some debug messages, set the ONION_DEBUG and/or ONION_DEBUG0 environment variable containing some source file names, e.g.

```
1 $ export ONION_DEBUG0='request.c url.c'
```

Dependencies

Required:

- C compiler
- cmake
- make
- One of:

-
- epoll (Linux)
 - libevent (Multiarch)
 - libev (Multiarch)

This compilers and minimum versions are known to work:

- Linux:
 - GCC 4.4
 - clang 3.0

For the C++ bindings a C++11 compiler is needed:

- Linux:
 - GCC 4.8
 - clang 3.7

Optional; Onion will compile but some functionality will not be available:

- gnutls and gcrypt (SSL support)
- pthreads (threading support)
- libxml2 (WebDAV support)
- libpam (HTTP Basic Auth support using PAM)
- C++ compiler
- Systemd (support for listening on systemd sockets)
- sqlite3 (sqlite3 session backend)
- hiredis (Redis session backend)

Optional for examples:

- cairo
- libpng2
- Boehm GC
- libjpeg

Semantic versioning.

Starting with Onion 0.8.0, we use semantic versioning, making the following promises:

- Version format is [MAJOR].[MINOR].[PATCH].
- Only make API and ABI changes at major versions. Can add functionalities at minor releases.
- Only remove API functions at major versions. No changes in semantics never.

-
- Minor versions are always ABI back-compatible. This means that if you compiled with a previous minor version and same major version, it will still compile and work.
 - Only add API functions at minor versions.
 - Only fix patches a patch revisions.

Patch revisions are is a non sequential number, so after 0.8.0 is not 0.8.1, but maybe 0.8.23. It will always increase.

Check [onion/version.h](#) for more information on version control.

SSL Support

If at compile time the build script finds the gnutls libraries, SSL support will be compiled in. It can be deactivated anyway at `./CMakeLists.txt`.

To use it you have to set the certificates, and you can check if its on, checking the flag `O_SSL_ACTIVATED`.

If support is not compiled in, then the library will not use SSL, but for the user of the library the interface is the same; it will only change that when trying to set the certificates it will fail. Anyway for clients its just to use the interface and they dont care at all if suport is in or not. No more than being able to use SSL.

This is not mandatory because there may be moments when the program's users do not want to support SSL for whatever reasons, for example speed.

Threads support

Currently there are two threading modes. It can be set so the server is created as threaded (`O_THREADED`), and it will create a new thread per connection. There is no data protection as on the listen phase there should not be any change to onion structures.

Nevertheless if new handlers are created they must set their own threading support as necessary.

It can be deactivated at `CMakeLists.txt`. If no `pthread` lib is found on the system, it is not compiled in.

Also when thread support is on, onion server can be set to work on another (non-main) thread. This is independant from `O_THREADED` operation; it can have one thread with your normal application and another thread that listens and processes web-requests. Its set with the `O_DETACH_LISTEN` flag. This is very useful when adding an extra web server to your application so it can be added without changes to the flow of your application, but you will need to thread protect your data if you access to it from the web server.

Finally there is a pool mode. Users can set a default number of threads (`onion_set_max_threads`), and using `epoll` the data is given to the threads. This is the highest performant method, with up to 30k web-requests served on a Intel(R) Core(TM)2 Duo CPU T6500 @2.10GHz.

Customizing low-level allocation and threads

Sometimes it may be needed to customize memory allocation and/or threads operation. This could be useful when using an alternative `malloc`, or if you wanted to use Hans Boehm's conservative garbage collector from <http://www.hboehm.info/gc/> e.g. if you use GNU guile. Then you need to define your own memory routines and pass them to `onion_low_initialize_memory_allocation` before any other calls to `onion`. Likewise, to customize threads operations, call `onion_low_initialize_threads`. See comments in header file `low.h`. A program using Onion and Boehm's GC should first define a memory failure routine which should never return:

```
1 /* the memory failure routine should never return! */
2 static void memory_failure(const char*msg) {
3     perror(msg);
4     exit(EXIT_FAILURE);
5 };
```

Then, your program (using both `onion` and Boehm's GC) should initialize both memory routines and threads, like:

```
1 onion_low_initialize_memory_allocation
2 (GC_malloc, GC_malloc_atomic, GC_calloc,
3  GC_realloc, GC_strdup, GC_free,
4  memory_failure);
5
6 onion_low_initialize_threads
7 (GC_pthread_create, GC_pthread_join,
8  GC_pthread_cancel, GC_pthread_detach,
9  GC_pthread_exit, GC_pthread_sigmask);
```

You might need to define your `GC_calloc` using `GC_malloc` and `memset` if your version of Boehm's GC don't provide it. After these low-level initialization you can use `Onion` as usual.

You could also want to call just `onion_low_initialize_threads` if you wanted to name threads created by the `onion` library (using `pthread_setname_np` on Linux) and/or change their priority (using `pthread_setschedprio`), etc.

ARM Support

It can be cross compiled for ARM directly from `cmake`. Just do:

```
1 $ mkdir arm
2 $ cd arm
3 $ cmake .. -DCMAKE_TOOLCHAIN_FILE=../toolchain/arm.txt
4 $ make
```

It needs the current system opack and otemplate to compile some examples, so if you want to use the examples on your instalation, compile and install libonion for the current system first.

Tested on ubuntu 10.10, with gcc-4.5-arm-linux-gnueabi and g++-4.5-arm-linux-gnueabi installed.

Templating support

Starting on 0.3.0 development onion has templating support via otemplate. It is a template system similar to django templates (<http://docs.djangoproject.com/en/dev/topics/templates/>).

Check more information on how to use them at [tools/otemplate/README.rst](#).

I18N

There is I18N support. Check wiki for details or `fileserver_otemplate` example.

Systemd

Systemd is integrated. If want to use it, just pass the flag `O_SYSTEMD` to the `onion_new()`.

Oterm has example socket and service files for oterm support.

FreeBSD/Darwin

Since september 2013 there is support for FreeBSD using libev or libevent. This work is not as tested as the Linux version, but if some compilation error arises, please send the bug report and we will fix it ASAP.

OSX/Darwin support is also available on the darwin branch.

Once this work stabilizes it will be merged back to master.

Environment variables

You can set the following environment variables -e.g. with the `export` builtin of `bash`- to modify run-time behaviour of onion:

-
- **ONION_LOG**
 - noinfo – Disables all info output to the console, to achieve faster results
 - nocolor – Disable color use by the log
 - nodebug – Do not show debug lines
 - syslog – Log to syslog. Can be changed programatically too, with the onion_log global function.
 - **ONION_DEBUG0** – Set the filename of a c source file, and DEBUG0 log messages are written. This is normally very verbose.
 - **ONION_SENDFILE** – Set to 0 do disable sendfile. Under some file systems it does not work. Until a detection code is in place, it can be disabled with this.

Binary compatibility breaks

We try hard to keep binary compatibility, but sometimes its hard. Here is a list of ABI breaks:

0.4.0

- Onion object private flags have moved. If on your code you rely on them, must recompile. If dont rely on them, everything should keep working.