carml Documentation

Release 18.3.0

meejah

Oct 30, 2018

Contents

1 Some Quick Examples **3**

2 License **5**

2.1 Installing carml 5
2.2 Tor Setup 6
2.3 General Options 6
2.4 The Subcommands 7
2.5 Development 15
2.6 HOWTOs 15
2.7 Releases 15

Note: This code is intended as utilities mostly to help developers and Tor experts interact with their systems. Nobody has audited it for anonymitiy leaks (or worse). Use at your own risk.

`carml` is a command-line tool to query and control a running Tor (including Tor Browser Bundle). You can do things like:

- list and remove streams and circuits;
- monitor stream, circuit and address-map events;
- watch for any Tor event and print it (or many) out;
- monitor bandwidth;
- run any Tor control-protocol command;
- pipe through common Unix tools like `grep`, `less`, `cut`, etcetera;
- download TBB through Tor, with pinned certs and signature checking;
- . . . even spit out and run `xplanet` configs (with router/circuit markers)!

It is written in Python and uses Tor's control-port via the txtorcon library.

Documentation at: carml.rtfd.org or carmlion6vt4az2q.onion/ **Code at:** github.com/meejah/carml

In some ways, `carml` started as a dumping-ground for things I happened to make Tor do at least once from Python code. Are there things you wish you could easily make Tor do from the command-line? File an enhancement bug at GitHub!

`carml` is also easy to extend, even with system- or virtualenv- installed packages.

Feedback is appreciated – pull-requests and bug-reports (including feature enhancements) welcome at GitHub or you can contact me in #tor-dev on OFTC or via *meejah at meejah dot ca* with the public-key contained in the source.

CHAPTER 1

Some Quick Examples

```
(venv)meejah@machine:~$ carml circ --list
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
Circuits:
   809: BUILT 29 minutes ago carmlfake0->~Unnamed->lobstertech
   810: BUILT 29 minutes ago ~carmelfake1->~toxiroxi->~SECxFreeBSD64
   811: BUILT 5 minutes ago carmelfake2->torpidsDEinterwerk->~rainbowwarrior
   813: BUILT 24 seconds ago carmlfake0->~arkhaios1->~IPredator
(venv)meejah@machine:~$ carml circ --delete 810
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
Deleting circuit "810"...
...circuit 172 gone.
(venv)meejah@machine:~$ echo "hello world" | carml pastebin --once
12 bytes to share.
Launching Tor: connected.
People using Tor Browser Bundle can find your paste at (once the descriptor uploads):

   http://ok2byooigb4v53be.onion

If you wish to keep the hidden-service keys, they're in (until we shut down):
/dev/shm/tortmp6eHPg4
Awaiting descriptor upload...
Descriptor uploaded; hidden-service should be reachable.
Mon Jul 21 13:54:38 2014: Serving request to User-Agent "curl/7.37.0".
Shutting down.
(venv3)meejah@machine:~$ carml tbb
Getting recommended versions from "http://expyuzz4wqqyqhjn.onion/projects/torbrowser/
↪RecommendedTBBVersions".
   7.5.5, 7.5.5-MacOS, 7.5.5-Linux, 7.5.5-Windows, 7.5.6, 7.5.6-MacOS,
   7.5.6-Linux, 7.5.6-Windows, 8.0a8, 8.0a8-MacOS, 8.0a8-Linux,
   8.0a8-Windows, 8.0a9, 8.0a9-MacOS, 8.0a9-Linux, 8.0a9-Windows
Note: there are alpha versions available; use --alpha to download.
Downloading "tor-browser-linux64-7.5.5_en-US.tar.xz.asc" from:
   http://rqef5a5mebgq46y5.onion/torbrowser/7.5.5/tor-browser-linux64-7.5.5_en-US.tar.
↪xz.asc
```

(continues on next page)

(continued from previous page)

```
Downloading "tor-browser-linux64-7.5.5_en-US.tar.xz" from:
   http://rqef5a5mebgq46y5.onion/torbrowser/7.5.5/tor-browser-linux64-7.5.5_en-US.tar.
↪xz
[    ] - 0.0 of 65.8 MiB (1s remaining)
[    ] - 6.6 of 65.8 MiB (153s remaining)
[   ] - 13.2 of 65.8 MiB (137s remaining)
[   ] - 19.8 of 65.8 MiB (120s remaining)
[  ] - 26.4 of 65.8 MiB (102s remaining)
[  ] - 32.9 of 65.8 MiB (85s remaining)
[ ] - 39.5 of 65.8 MiB (70s remaining)
[ ] - 46.1 of 65.8 MiB (55s remaining)
[] - 52.7 of 65.8 MiB (38s remaining)
[] - 59.3 of 65.8 MiB (19s remaining)
[] - 65.8 of 65.8 MiB (0s remaining)
0.32 MiB/s
gpg: assuming signed data in 'tor-browser-linux64-7.5.5_en-US.tar.xz'
gpg: Signature made Sat 09 Jun 2018 06:42:37 AM MDT
gpg:                using RSA key D1483FA6C3C07136
gpg: Good signature from "Tor Browser Developers (signing key) <torbrowser@torproject.
↪org>" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EF6E 286D DA85 EA2A 4BA7  DE68 4E2C 6E87 9329 8290
     Subkey fingerprint: A430 0A6B C93C 0877 A445  1486 D148 3FA6 C3C0 7136
Signature is good.
Extracting "tor-browser-linux64-7.5.5_en-US.tar.xz"...
  decompressing...
   20% extracted
   40% extracted
   60% extracted
   80% extracted
  100% extracted
Tor Browser Bundle downloaded and extracted.
running: ./tor-browser_en-US/Browser/start-tor-browser
```

CHAPTER 2

License

carml is public domain. See unlicense.org for more information.

2.1 Installing carml

Note (for PyPI or development installs) you'll need to install libffi and liblzma development libraries. How to do this on various architectures (please send missing ones!):

- Debian + Ubuntu: apt-get install build-essential python-dev python-virtualenv libffi-dev liblzma-dev.

2.1.1 PyPI

Once you have libraries installed as above, you should be able to do a simple pip install carml. It's also possible to point to the .whl file (e.g. after signature verification).

It is recommended to use virtualenv to try without affecting system packages:

```
virtualenv venv
. ./venv/bin/activate
pip install carml
```

2.1.2 Development/Source

From a fresh clone (git clone https://github.com/meejah/carml.git) type make venv. Then activate your new virtualenv with source ./venv/bin/activate and then pip install --editable . which should install all the dependencies (listed in requirements.txt).

To do this and use peep, you need pip version 6.1.1. So, you you can try something like this (from the root of a fresh clone):

```
virtualenv venv
. ./venv/bin/activate
pip install --upgrade pip setuptools  # esp. for Debian
pip install --editable .
```

Dependencies:

- txtorcon
- humanize
- ansicolors
- PyOpenSSL
- txsocksx
- backports.lzma

2.2 Tor Setup

For Tor setup, make sure you have at least the following in `/etc/tor/torrc`:

```
CookieAuthentication 1
CookieAuthFileGroupReadable 1
ControlPort 9051
# corresponding carml option: "--connect tcp:127.0.0.1:9051"
```

Or, if you prefer Unix sockets (recommended):

```
CookieAuthentication 1
ControlSocketsGroupWritable 1
ControlSocket /var/run/tor/control
# corresponding carml option: "--connect unix:/var/run/tor/control"
```

The port or unix-socket can obviously be whatever; the above are Tor's defaults on Debian. The Tor Browser Bundle defaults to using 9151 for the control socket (and DOES use cookie authentication by default).

On Debian/Ubuntu you need to be part of the `debian-tor` group. To check, type `groups` and verify `debian-tor` is on the list. If not, add yourself (as root, do):

```
# usermod username --append --groups debian-tor
```

If you changed Tor's configuration, don't forget to tell it (as root):

```
# service tor reload
```

2.3 General Options

The `carml` command itself takes a few options that are common to all sub-commands.

2.3.1 --connect, -c

How to connect to Tor. This accepts a Twisted endpoint client string as well as just a port. The default is `localhost:9151` (Tor Browser Bundle default). Some examples:

```
$ carml --connect 9151
$ carml --connect 127.0.0.1:9051
$ carml --connect tcp:port=9051:host=127.0.0.1
```

If you use password authentication, you can supply one with `--password` or `-p`. If you're on the same machine, use cookie authentication instead.

2.3.2 --quiet, -q

As little output as possible on standard out. Warnings may still be printed on standard error.

2.3.3 --info, -i

Print Tor version when we connect, and whether it is dormant or not.

2.3.4 --color, C

Whether to output colors or not. Can be `auto` (the default), `no` or `always`. You can also use the separate option `--no-color` which is the same as `--color=no`

2.3.5 --timestamps, -t

Prepend messages with a timestamp.

2.3.6 --debug, -d

If there's an error, print the stack trace out along with the error message; could be useful for bug-reports and development.

2.4 The Subcommands

Similar to programs like `git`, the real functionality of carml is in the sub-commands. They all take their own options (but obey global options listed above). You can get any help on a command with the `help` subcommand, like: `carml help subcommand`

2.4.1 onion

This command starts an Onion Service on the connected Tor whose livetime is tied to this process (when *carml onion* exits, the Onion Service will be removed from Tor).

You may forward more than one port which might be useful for e.g. an HTTP and Git server on the same Onion address. The "local" portion may include an IP address and port or a unix-domain socket.

Examples

Forward both "80" and "9418" on a version 3 service. This will forward an incoming request on port 80 to 127.0.0.1:80 (and 9418 to 127.0.0.1:9418) and is the simplest way to set up a service:

```
carml onion --port 80 --port 9418
```

If you are running the other services in containers or similar, perhaps they are on different (but still local) IP addresses. Note that we're also forward "80" to "8888" here, which is fine:

```
carml onion --port 80:192.168.0.123:8888 --port 9418:192.168.0.123:9418
```

You may also run services via unix sockets. If your other services are on the same machine, this is the safest and fastest way to forward:

```
carml onion --port 80:unix:/var/run/nginx_sock --port 9418:unix:/var/run/git_sock
```

2.4.2 `pastebin`

This launches a new hidden-service to share some data as the `text/plain` MIME type via a Twisted Web server. By default, the data to share is read from stdin. You may also use the option `--file` (`-f`) to share a single file instead.

To use stealth authentication on your hidden-service, you can pass the `--keys` (`-k`) option which specifies how many authentication cookies to create. This will print out the commands you can send (**securely**!) to the people you want to share with.

If you wish to serve an entire hierarchy of files as a Web site, instead see instructions at txtorcon (would look like `twistd web --port "onion:80" --path ~/public_html` with txtorcon installed).

A similar alternative is also onionshare for diverse file-types on many OSes. OnionShare also comes with a GUI.

Note: Note that the hidden-service private keys are in a freshly created temporary directory (`TMPDIR` is honoured) and that you must **save them yourself (by copying them somewhere)** before you end (e.g with Control-C) the `carml pastebin` command, which deletes the tempdir (including the new keys).

If you want to see what it will look like to the people you're sharing the link with, use `--dry-run` (`-d`) which starts a local listener only (i.e. doesn't launch a Tor, nor set up an actual hidden service). This is preferable to actually-launching a service just to test it.

Examples

```
$ export TMPDIR=/dev/shm
$ echo "hello hidden-serice world" | carml pastebin
25 bytes to share.
Launching Tor: connected.
People using Tor Browser Bundle can find your paste at (once the descriptor uploads):

   http://ok2byooigb4v53be.onion

If you wish to keep the hidden-service keys, they're in (until we shut down):
/dev/shm/tortmp6eHPg4
Awaiting descriptor upload...
Descriptor uploaded; hidden-service should be reachable.
```

(continues on next page)

(continued from previous page)

```
Mon Jul 21 13:54:38 2014: Serving request to User-Agent "curl/7.37.0".
^CShutting down.
$
```

If you used the stealth-authentication version, it might look like this:

```
$ carml pastebin -f README.rst --keys 5
4573 bytes to share with 5 authenticated clients.
Launching Tor.
[          ] Connecting to directory server
[         ] Finishing handshake with directory server
[         ] Establishing an encrypted directory connection
[        ] Asking for networkstatus consensus
...
[  ] Loading relay descriptors
[  ] Loading relay descriptors
[ ] Connecting to the Tor network
[] Establishing a Tor circuit
[] Done
[] Waiting for descriptor upload...
[] At least one descriptor uploaded.
You requested stealth authentication.
Tor has created 5 keys; each key should be given to one person.
They can set one using the "HidServAuth" torrc option, like so:

  HidServAuth ww2ufwkgxb2kag6t.onion ErQPDEHdNNprvWYCA2vTLR
  HidServAuth f5kb64pe3nygyplx.onion HeemYe0TIoOzU/WkjJwP3R
  HidServAuth ywhbfzepvss5hecm.onion 8JcZKcS8YQXMuYBF/G1z8x
  HidServAuth pow2d55j6ezrruib.onion jK6/yXZ2R7xDsf3sm/PyVh
  HidServAuth t7gnlwzw4hjxc45z.onion ezUZBaPmFYSzrGeZXYJfGh

Alternatively, any Twisted endpoint-aware client can be given
the following string as an endpoint:

  tor:ww2ufwkgxb2kag6t.onion:authCookie=ErQPDEHdNNprvWYCA2vTLR
  tor:f5kb64pe3nygyplx.onion:authCookie=HeemYe0TIoOzU/WkjJwP3R
  tor:ywhbfzepvss5hecm.onion:authCookie=8JcZKcS8YQXMuYBF/G1z8x
  tor:pow2d55j6ezrruib.onion:authCookie=jK6/yXZ2R7xDsf3sm/PyVh
  tor:t7gnlwzw4hjxc45z.onion:authCookie=ezUZBaPmFYSzrGeZXYJfGh

For example, using carml:

  carml copybin --onion tor:ww2ufwkgxb2kag6t.onion:authCookie=ErQPDEHdNNprvWYCA2vTLR
  carml copybin --onion tor:f5kb64pe3nygyplx.onion:authCookie=HeemYe0TIoOzU/WkjJwP3R
  carml copybin --onion tor:ywhbfzepvss5hecm.onion:authCookie=8JcZKcS8YQXMuYBF/G1z8x
  carml copybin --onion tor:pow2d55j6ezrruib.onion:authCookie=jK6/yXZ2R7xDsf3sm/PyVh
  carml copybin --onion tor:t7gnlwzw4hjxc45z.onion:authCookie=ezUZBaPmFYSzrGeZXYJfGh
```

2.4.3 `copybin`

This command downloads the contents of a `carml pastebin`.

Mostly it exists because there's not a good way to specify stealth-authentication cookies to commands like `curl` or similar.

The only option is `--service` which is a Twisted endpoint string describing the service. Note that the "tor:..."

string can be used with any Twisted program that uses client endpoint strings (see clientFromString).

This is **experimental**: you'll need to get a `txtorcon` from the `stealth-authentication` branch; `pip install -e git+https://github.com/meejah/txtorcon.git@stealth-authentication#egg=txtorcon`)

This will end up looking something like this:

```
$ carml copybin -s tor:ccsq7wrrm2ejkhmg.onion:authCookie=POYAUUZf4O28iJM0ZpIiwx
```

2.4.4 `downloadbundle`

Note: This command requires the optional txsocksx library to be installed. Simply a `pip install txsocksx`

The `downloadbundle` command figures out what the latest Tor Browser Bundle is (from check.torproject.org), downloads the package for your operating system and (optionally) extracts it. It has bundled certificates for torproject.org and **checks that the public keys** are the same. It also **checks the signature** on the downloaded bundle, using bundled keys for Tor people or (optionally) the current user's GnuPG keychain.

To use your own keychain, use `--system-keychain` (`-K`). By default, the command builds a tempdir for GnuPG and imports the bundled keys (of Tor people who typically sign the release) there.

Use `--beta` (`-b`) to download the latest Beta release instead (if available).

Use `--no-extract` (`-E`) if you do not wish to extract the bundle after downloading. You additionally need `backports.lzma` installed for this to work.

If you're really feeling adventurous, don't have a system Tor running, or can't install `txsocksx` for some reason, you can (completely inadvisably) pass `--use-clearnet` to download over the plain Internet. Of course, you still get the certificate pins and signature checking.

Examples

```
$ carml downloadbundle -e
Getting recommended versions from "https://check.torproject.org/RecommendedTBBVersions
↪".
   3.6-Linux, 3.6-MacOS, 3.6-Windows, 3.6.1-Linux, 3.6.1-MacOS,
   3.6.1-Windows
tor-browser-linux64-3.6.1_en-US.tar.xz.asc: already exists, so not downloading.
tor-browser-linux64-3.6.1_en-US.tar.xz: already exists, so not downloading.
gpg: Signature made Tue 06 May 2014 05:37:07 PM MDT using RSA key ID 63FEE659
gpg: Good signature from "Erinn Clark <erinn@torproject.org>"
gpg:                 aka "Erinn Clark <erinn@debian.org>"
gpg:                 aka "Erinn Clark <erinn@double-helix.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 8738 A680 B84B 3031 A630  F2DB 416F 0610 63FE E659
Signature is good.
Extracting "tor-browser-linux64-3.6.1_en-US.tar.xz"...
  decompressing...
   20% extracted
   40% extracted
   60% extracted
   80% extracted
```

(continues on next page)

(continued from previous page)

```
  100% extracted
Tor Browser Bundle downloaded and extracted.
To run:
   ./tor-browser_en-US/start-tor-browser
```

Note that for users who have a valid trust-path to Erinn Clark, using `--system-keychain` would avoid the WARNING: from GnuPG.

2.4.5 monitor

This command will print out information about circuits, streams and/or address mappings, and continue listening for circuit and stream events. If you just want the current state, use `--once` to exit after the initial state is dumped out.

If you don't want circuits, pass `--no-circuits` (`-c`). Similarily, there are `--no-streams` (`-s`), `--no-guards` (`-g`) and `--no-addr` (`-a`) options.

For even more information, `--verbose` (`-v`).

You can also include log messages by passing `--log-level=INFO` (`-l`).

Examples

```
$ carml monitor
$ carml monitor --no-guards --log-level=WARN
$ carml monitor -sga
```

2.4.6 stream

This command is the sister of `carml circ`, allowing you to view and play with streams.

Currently, you can do one of three things:

- `--list` (`-L`) shows you all current streams
- `--attach` (`-a`) forces all subsequent streams to attach to a particular circuit-id (until you exit carml with Control-C)
- `--close` (`-d`) close a stream

Examples

```
$ carml circ -L
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
Circuits:
   974: BUILT 14 minutes ago carmlfake0->fluxe4->~TorLand1
   975: BUILT 14 minutes ago carmlfake1->bethesdatech->Dontbleed2
$ carml stream --attach 975
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
Exiting (e.g. Ctrl-C) will cause Tor to resume choosing circuits.
Attaching all new streams to Circuit 975.
    carmlfake1->bethesdatech->Dontbleed2
  attaching 1719 (resolve encrypted.google.com)
  attaching 1720 encrypted.google.com:443
```

(continues on next page)

(continued from previous page)

```
^C
$
```

2.4.7 xplanet

This command spits out valid xplanet configuartion files for the "marker" files. If you pass the `--execute` (`-x`) argument, a tempdir is created with a top-level xplanet configuration and xplanet is run against that (causing a map to appear on your root window).

You can use `--all` (`-A`) to output markers for ALL routers (instead of just ones active for you right now). Note that the position of many will overlap as we don't do anything smart when two co-ordinates are identical.

xplanet can include an `arc_file` for drawing lines between relays. With `-x` or `-f` this is used to draw links between relays in a circuit. You can also use `--arc-file` (`-a`) if you're not using `-x` or `-f`.

Warning: Obviously, this could easily leak some information about which relays and circuits you are currently using. Since your guard-nodes (first hop of a circuit) are long-lived, it's advisable to use this for entertainment purposes mainly, and clear your root window when done.

Examples

```
$ carml xplanet -f
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
3 (23%) routers with no geoip information.
xplanet -num_times 1 -projection rectangular -config /tmp/tmp_32oE5/xplanet-config
<Circuit 977 BUILT [redacted IPs] for GENERAL>
<Circuit 977 BUILT [redacted IPs] for GENERAL>
4 (27%) routers with no geoip information.
xplanet -num_times 1 -projection rectangular -config /tmp/tmp_32oE5/xplanet-config
<Circuit 978 BUILT [redacted IPs] for GENERAL>
<Circuit 978 BUILT [redacted IPs] for GENERAL>
5 (29%) routers with no geoip information.
xplanet -num_times 1 -projection rectangular -config /tmp/tmp_32oE5/xplanet-config
```

2.4.8 cmd

The command named `cmd` takes the rest of the command-line and sends it straight to Tor as a control-protocol request (see the torspec repository for full details). It then prints out the reply from Tor. (This isn't really suitable for events; see the `events` command).

If you pass a single dash as the command-line (that is, `carml cmd -`) then commands are read one line at a time from stdin and executed sequentially.

Examples

```
$ carml -q cmd getinfo info/names | tail -5
status/version/recommended -- List of currently recommended versions.
stream-status -- List of current streams.
traffic/read -- Bytes read since the process was started.
```

(continues on next page)

(continued from previous page)

```
traffic/written -- Bytes written since the process was started.
version -- The current version of Tor.

$ carml -q cmd SIGNAL NEWNYM
OK

$ echo "getinfo net/listeners/socks" > commands
$ echo "getinfo traffic/read" >> commands
$ echo "getinfo traffic/written" >> commands
$ cat commands | carml -q cmd -
Keep entering keys to run CMD on. Control-d to exit.
net/listeners/socks="127.0.0.1:9050"
traffic/read=6667674
traffic/written=391959

$ carml -q cmd getinfo net/listeners/socks traffic/read traffic/written
net/listeners/socks="127.0.0.1:9050"
traffic/read=10012841
traffic/written=516428
```

2.4.9 `circ`

Play with your circuits. You can do a few main actions:

- `--list` (`-L`) list the current circuits (similar to `carml monitor`).
- `--build` (`-b`) build a new circuit, either specifying relays by hand or "auto" to let Tor select. You may also use a `*` as a stand-in for any positional circuit; only Guards will be selected for the first one.
- `--delete` to delete a circuit (pass `--if-unused` or `-u` to only delete it after it's no longer used).

The ~ characters in the names means that router doesn't have the "Named" flag.

Examples

```
$ carml circ --build *,*,AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Building circuit: ~whatever->~somerandomrouter->~router_with_all_As
Circuit ID 83: ~whatever->~somerandomrouter->~router_with_all_As: built.
$ carml circ --build auto
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
Building new circuit, letting Tor select the path.
Circuit ID 982: carmlfake0->ryro->nationalliberal: built.
$ carml circ --build nationalliberal,ryro,nationalliberal
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
Circuit ID 983: nationalliberal->ryro: failed (DESTROYED, TORPROTOCOL).
$ carml circl --list
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
Circuits:
   977: BUILT 10 minutes ago ~carmlfake0->kasperskytor01->~Unnamed
   978: BUILT 10 minutes ago ~carmlfake0->~Unnamed->persladange2
   982: BUILT a minute ago ~carmlfake0->ryro->~nationalliberal
```

2.4.10 newid

This basically just runs `signal newnym` (which you could do with *cmd* of course) but also verifies that Tor actually does give you a new identity (which can fail, as this command is rate-limited).

Usage is simple:

```
$ carml newid
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
Requesting new identity
success.
```

2.4.11 events

Simplistic interaction with Tor's "Events". This simply subscribes to the event(s) you list, and prints out the text Tor sends back.

If you only want to listen for a certain number of events, use `--count` (`-n`) with an argument or the special-case `--once` for a single event. This might be useful, for example, to determine when your Tor downloads a new consensus (like the first example, but use NEWCONSENSUS instead).

Note that the count of events is global; if you listen for 2 different events with `--once`, the command will exit after the first event (i.e. not one of each).

Examples

```
$ carml -q events --once ADDRMAP
carml.readthedocs.org 162.209.114.75 "2014-06-04 23:47:37" EXPIRES="2014-06-05␣
↪05:47:37" CACHED="YES"
$ carml events --count 5 INFO
Connected to a Tor version "0.2.4.21 (git-c5a648cc6f218339)" (status: recommended).
exit circ (length 3): carmlfake0(open) carmlfake1(open)
↪$AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA(open)
pathbias_count_use_attempt(): Used circuit 982 is already in path state use succeeded.
↪ Circuit is a General-purpose client currently open.
link_apconn_to_circ(): Looks like completed circuit to [scrubbed] does allow␣
↪optimistic data for connection to [scrubbed]
connection_ap_handshake_send_resolve(): Address sent for resolve, ap socket 14, n_
↪circ_id 2147503826
connection_edge_process_inbuf(): data from edge while in 'waiting for resolve response
↪' state. Leaving it on buffer.
```

2.4.12 relay

List and find relays. These are based on the current notion of the consensus that the Tor we've connected to has. This comes from "microdescriptors" that Tor downloads from Directory Authorities periodically.

Relays are usually referred to by their "hex ID", a 40-character representation of the actual (binary) relay ID which itself is a hash of the relay's public identity key.

Use `carml relay --info` to search for a relay by key-ID or its name (or a subset thereof) and print some information about the relay (or relays) found.

Sometimes relays can come and go; if you want to wait for a relay with a particular hex-ID to be in the consensus, use `carml relay --await hex_id`. This will either work immediately (if the relay is already in the consensus) or wait for `NEWCONSENSUS` events to see if the relay has appeared yet.

Examples

```
$ carml relay --list | wc -l
7197
$ carml relay --info [hex id]
     name: [redacted]
   hex id: $[redacted]
 location: XX
  address: [redacted]:9011 (DirPort=9030)
 last published 28 days ago
```

2.5 Development

Write me.

2.6 HOWTOs

Pull-requests with other ideas or instructions are encouraged!

2.6.1 Ensure Your Relay Is In Consensus

Let's say you wanted to ensure that your relay is in each consensus that's published (and didn't wish to use the Tor Weather web-service). You could use the *events* sub-command to wait for a NEWCONSENSUS event and `grep` through it for your relay's fingerprint.

Wrapping this in a `while` loop would make it run forever.

2.7 Releases

2.7.1 master

2.7.2 18.3.0

- October 23, 2018
- fix a few Python3 issues
- OnionOO changed a few result tags
- *onion* can accept + output private-keys
- fix bug with *carml onion* not accepting single-port *–port* correctly

2.7.3 18.2.0

- September 26, 2018
- *carml* is now Python3-only
- add *onion* sub-command to add Onion services to a Tor instance

2.7.4 18.1.0

- June 26, 2018

2.7.5 18.0.0

- April 16, 2018

2.7.6 17.4.0

- May 15, 2017

2.7.7 17.3.0 and earlier

- see PyPI

2.7.8 0.0.0 - 0.0.6

- August 3, 2014
- Initial release. Some issues with PyPI caused my to have to incrememnt the version several times.